Generating DVR Data Model for Data Warehouse

by

Mr. Paisarn Trakulsuk

A Doctoral Dissertation

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in Computer Information Systems
Assumption University

October, 2000

Generating DVR Data Model for Data Warehouse

by
Paisarn Trakulsuk

A Doctoral Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of Doctor Philosophy
in Computer Information Systems

Dissertation Committee: Prof.Dr. Srisakdi Charmonman (Chairperson)
Assoc.Prof. Somchai Thayarnyong
Asst.Prof.Dr. Vichit Avatchanakorn (Advisor)
Air Marshal Dr. Chulit Meesajjee
Dr. Chamnong Jungthirapanich
Asst.Prof.Dr. Ouen Pin-ngern
Assoc.Prof.Dr. Suphamit Chittayasothorn
Dr. Boonyarit Phokrud

Name : Mr. Paisarn Trakulsuk
Nationality : Thai
Previous degree : B.B.A. (Business Computer),
Assumption University

M.S. (CIS), Assumption University

Assumption University
Bangkok, Thailand
October 2000

Ph.D. Program
Dissertation

Generating DVR Data Model for Data Warehouse

Submitted to: Prof.Dr. Srisakdi Charmonman (Chairperson)
Assoc.Prof. Somchai Thayarnyong
Asst.Prof.Dr. Vichit Avatchanakorn (Advisor)
Air Marshal Dr. Chulit Meesajjee
Dr. Chamnong Jungthirapanich
Asst.Prof.Dr. Ouen Pin-ngern
Assoc.Prof.Dr. Suphamit Chittayasothorn
Dr. Boonyarit Phokrud

Submitted on: October 2000

Submitted by: Paisarn Trakulsuk

A Doctoral Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy
in Computer Information Systems

Assumption University
Bangkok Thailand
October 2000

Generating DVR Data Model for Data Warehouse

by
Mr. Paisarn Trakulsuk

A Doctoral Dissertation

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in Computer Information Systems
Assumption University

October 2000

RESEARCH TITLE	:	Generating DVR Data Model for Data Warehouse
CANDIDATE NAME	:	Mr. Paisarn Trakulsuk
ADVISOR NAME	:	Asst.Prof.Dr.Vichit Avatchanakorn
ACADEMIC YEAR	:	October 2000

The Graduate School of Assumption University had approved this final dissertation as a partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Information Systems.

(Prof.Dr. Srisakdi Charmonman)
Chairperson of Examination Committee

(Assoc.Prof.Somchai Thayarnyong)
MUA Representative

(Asst.Prof.Dr.Vichit Avatchanakorn)
Advisor

(Air Marshal Dr.Chulit Meesajjee)
Member

(Dr.Chamnong Jungthirapanich)
Member

(Asst.Prof.Dr.Ouen Pin-ngern)
Member

(Assoc.Prof.Dr. Suphamit Chittayasothorn)
Member

(Dr.Boonyarit Phokrud)
Member

ABSTRACT

The valuable information from multiple data sources in warehouse database is summarized in form of multidimensional data model, generally implemented in relational database. This research presents a procedure to generate the Dimension, Variable, and Relative Dimension of multidimensional data model used in the Multidimensional On-Line Analytical Processing (MOLAP) concept.

The proposed model is presented in two concepts: (1) Building DVR model of warehouse database, the proposed procedure consists of classification, partitioning, and clustering modules. (2) The DVR data model design using graph model, the proposed design shows how to provide data support intelligence through multidimensional data analysis when there are huge amount of online data. This design also represents a fact scheme that integrates relative information and external sources, as well as extends the algorithms to build DVR patterns of multidimensional data model.

ACKNOWLEDGEMENTS

Several People have made contribution to this research. The writer would like to acknowledge their efforts and thank them for their contributions.

He would like to gratefully acknowledge to parent who motivated him for studying Ph.D. Program in Assumption University. In Particular he would like to thank Asst.Prof.Dr. Vichit Avatchanakorn, his project advisor, who gave the idea and advice given in to preparation of this research.

He would like to thank to Prof.Dr. Srisakdi Chamonman, chairman of Ph.D. Program, and the committees, Air Marshal Dr. Chulit Meesajjee, Assoc.Prof. Somchai Thayarnyong, Asst.Prof.Dr. Ouen Pin-ngern, Assoc.Prof.Dr. Suphamit Chittayasothorn, Dr. Chamnong Jungthirapanich, and Dr. Boonyarit Phokrud for their valuable suggestions and shared the ideas of significant mentions to develop his research.

He extend his sincere thank to Mr. Jaime Cabrara who helped him to check and prove the contents and grammar of his research. Lastly he would like to thank you the reader, and hope you find the content of its useful and easy to read.

NOMENCLATURE

$\forall x$	For any attribute x, …
I_x	Index key of attribute x
δ_x	Data Property or Characteristics of attribute x
$\cup$	Union, Or
D_x^i	Dimension from attribute x of data entity i
V_x^i	Variable from attribute x of data entity i
$\overline{A}$	The mapped data set of A
∞	Join Function (e.g. $a_1 \infty a_2$ is attribute a_1 joined with attribute a_2.)
$\leftrightarrow$	Related Function.
$\Rightarrow$	Associated Function.
Γ_m^n	Query process in which m represents the *SQL* aggregate functions (i.e. count, sum, max, min etc.) and n is grouped according attribute.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

I. INTRODUCTION

1.1 Introduction

The database system, one of the most essential aspects of computational sciences, is applied in various fields such as commerce, education, arts, and the sciences. Since 1946 when the first mainframe ENIAC was created, the approaches to manipulating databases developed in many ways. In the 70's, IBM introduced a significant development in early database management systems, the data management component of the Information Management System (IMS). Dr. E. F. Codd's 1970 model for managing data, the relational model, was followed by commercial relational DBMS's, even as IBM developed the hierarchical database in IMS systems. The hierarchical data structure is a parent-child relationship between pairs of record types. A hierarchical database is defined as a hierarchical schema or a definition tree [134]. Microcomputer DBMS's developed in 1980, due to increased systems functionality and new data structures, most interesting of which are the Multidimensional Database (MDDB) and Data Warehouse (DW).

Throughout the history of system development, there are two fundamental requirements: operational and analysis systems. The operational systems need performance whereas analysis systems need flexibility and broad scope. Today's data warehousing systems provide the analytical functions that are most successful designing with the overall business structure rather than specific requirements. This significant influence on evolution of data warehousing science is the fundamental changes in business organization and structure during late eighties and early nineties.

1.2 Overview of Data Warehousing

The data warehouse is a collection of integrated, subject-oriented database systems designed to support the decision support systems (DSS) function. [20]. Inmon redefined the data warehouse as a "subject-oriented, integrated, time-variant, non-volatile collection of data to support managerial decision-making."

As a collection of computer-based information, the data warehouse is critical in the success of enterprise initiatives. Designed for enterprise-wide access, it deviates from the more classical process orientation of applications organized by operational systems. While most operational systems are designed around specific applications and functions such as inventory, payroll, and human resources, the data warehouse is organized around broader subjects such as customers, suppliers, products, and activities.

Data flow from original sources to the data warehouse involves access, transformation, distribution, and storage in relational data form [172] (Figure 1.1), with operational and external databases as sources of historical and up-to-date data. The DSS analyst relies on such information to access the data warehouse layer.

Data access (also called Data access layer) is the process of reading, seeking, or writing data on a storage unit. On the other hand, data transformation (also known as data scrubbing or data cleansing) is creating information from data by decoding data and merging records from multiple DBMS formats. Data transformation and distribution are executed whenever the warehouse data is updated. This involves selection, editing, summarizing, combining and loading data from operational and/or external database. In data transformation, complex programming processes are used: data analysis, filtering, identifying patterns, and data structuring. To facilitate the process, data warehousing tools are also created.

In the data warehouse layer, data is primarily used for information. In a physical data warehouse, copies of operational and external data are stored in highly flexible and accessible forms. Increasing volume of the data warehouse demands summarization of data in multidimensional view patterns for archiving. Summarizing or transforming data may use additional modules or programs to build the multidimensional data structure, which is designed to be usable to other data levels.

To sum up, storing information effectively in the data warehouse entails much work, specialized programming, and recording in metadata all data descriptions, types, sources, and other pertinent information. The goal is interoperability, which is vital to data warehousing.

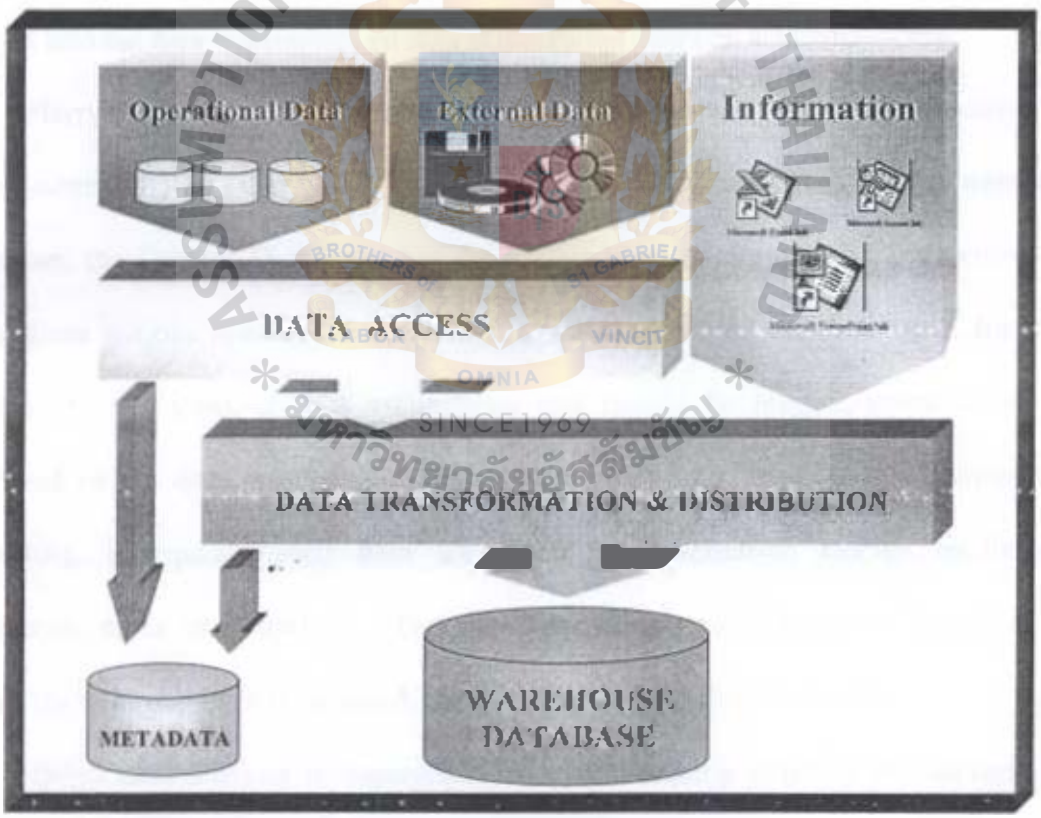


Figure 1.1. Data Warehouse Architecture.

The data warehouse system integrates operational data from various sources into a single and coherent structure. The goal is to support analysis and decision-making processes within an enterprise. An accurate data warehouse allows time-variant data to show up in key structures such as day, week, and month. Because data in the data warehouse do not change, loading data from operational systems into the warehouse database is easily done.

There are different ways to integrate data sources of operational systems. For example, the SEX attribute may be represented in different encoding applications: as Male and Female in Application A, 'M' and 'F' in Application B, '0' and '1' in Application C, and, 'x' and 'y' in Application D. However, when the SEX attribute is loaded into the data warehouse, all data is converted into one standard format.

Harry Singh's Data Warehouse Concepts classifies data warehouse structures into four: current data, older data, summarized data, and metadata. In the warehouse structure, the Current Data level has the lowest level of granularity. Data enters this level from various operational applications, and are mostly stored in disks for faster access. At this level, data is voluminous and should be purged, summarized, and archived to the data warehouse. Disk storage should be very large. To avoid data crowding, infrequently-used data are stored on alternative storage media (i.e., cartridges, tapes or diskettes), called the Older Data level. To access data from this level, the storage media is reloaded, linked, or coupled to the user's unit.

Quick data analysis in management decision-making requires summarized data. There are two types: lightly summarized and highly summarized data. The former is a compiled set of data (e.g. sales volume of product group during January to December) while the latter is more compact, accessible, and usually stored in the data warehouse. Outside data sources may be used for high-performance analysis (i.e., standard man-

hour maintenance costs, average cost of repaired engines among top five manufacturers).

Metadata identifies data in the data warehouse [18]. It includes data contents, structures, formulas, calculations, sources, and locations. It is the data directory that helps DSS analysts find data. It is a mapping guide for warehousing operational data.

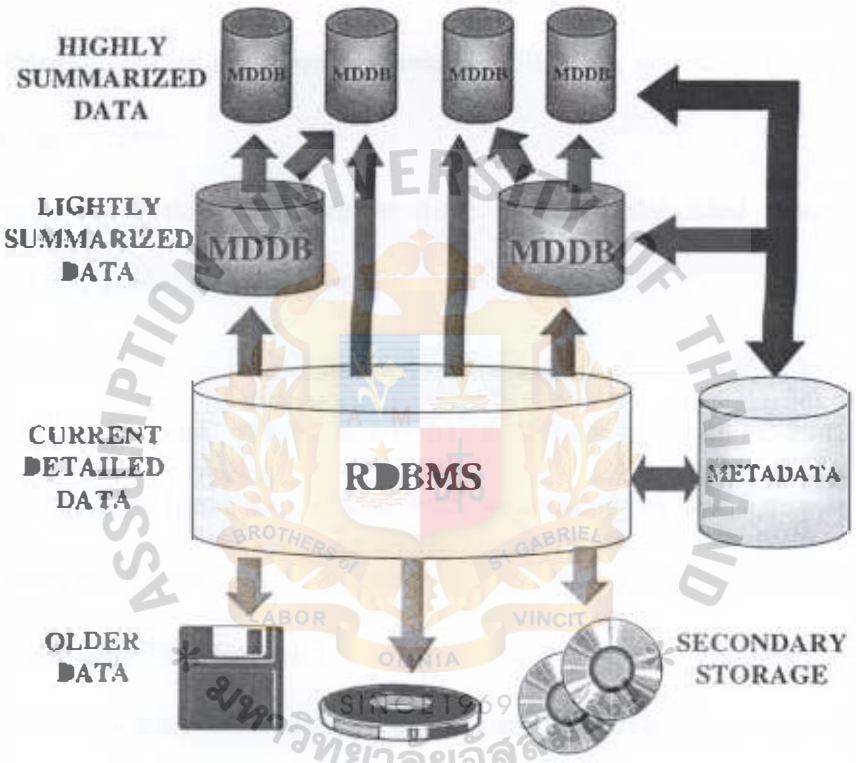


Figure 1.2. Data Warehouse Structure.

Figure 1.2 illustrates the levels in a data warehouse structure and the database types used in each level. Current detailed data are occasionally stored in a relational database that allows the management of large volumes of data. Data access is possible through high-performance query tools or other statistical application software.

Highly-summarized and lightly-summarized data are stored in a multidimensional format in a relational or a multidimensional database. The multidimensional database is designed for efficient data management in

multidimensional view patterns, widely used today. Older data are stored in tapes or diskettes.

1.3 Overview of Multidimensional Databases

To improve performance analysis, data must be viewed in multidimensional patterns. The database may store data in either relational or multidimensional formats. Although both can be viewed in multidimensional patterns, the relational database is is difficult to translate data formats into hypercube structures, and cannot support the drill-down or roll-up methods.

Hypercube is a data structure in three or more flat-sided dimensions, each dimension at right angles to the others.

The Multidimensional Database (MDDB) is a computer software system designed for efficient and convenient storage and retrieval of large volumes of data. Hypercube patterns allow the fastest data access. It also allows rotating the data figure as desired.

For example, Figure 1.3 illustrates various perspectives of the consumption volume of three specific spare parts used by three engines in three types of Boeing aircraft. Here, data is stored in three-dimensional array patterns (hypercubes).

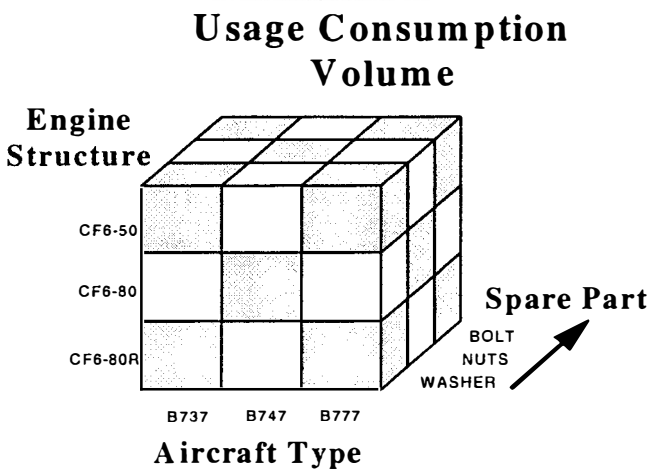


Figure 1.3. Multidimensional Database Model Sample.

Multidimensionality refers to a database technology that enables faster data analysis. The spreadsheet format helps users to view data in various perspectives.

A multidimensional database is easier and faster to use than a relational database. A relational database is organized around a list of records, each containing related information that is organized into fields or attributes. On the other hand, a multidimensional database records data as arrays, each composed of at least one dimension and fact or measure. These are represented in patterns of at least two dimensional matrices.

ENGINE STRUCTURE		
PART	ENGINE	QTY
BOLT	CF6-50	90
BOLT	CF6-80	120
BOLT	CF6-80R	140
NUTS	CF6-50	50
NUTS	CF6-80	60
NUTS	CF6-80R	100
SCREWS	CF6-50	40
SCREWS	CF6-80	70
SCREWS	CF6-80R	80
WASHER	CF6-50	20
WASHER	CF6-80	10
WASHER	CF6-80R	30

ENGINE STRUCTURE			
PART	CF6-50	CF6-80	CF6-80R
BOLT	90	120	140
NUTS	50	60	100
SCREWS	40	70	80
WASHER	20	10	30

Figure 1.4. Relational vs Multidimensional Data Presentation.

Figure 1.4 shows data arranged in relational and multidimensional formats. The relational format presents data in three modules while the multidimensional format employs a two-dimensional matrix. The latter gives a , hence a. A two-dimensional matrix is a more fluid data structure.

ENGINE STRUCTURE		
PART	ENGINE	QTY
BOLT	CF6-50	90
BOLT	CF6-80	120
BOLT	CF6-80R	140
NUTS	CF6-50	50
NUTS	CF6-80	60
NUTS	CF6-80R	100
SCREWS	CF6-50	40
SCREWS	CF6-80	70
SCREWS	CF6-80R	80
WASHER	CF6-50	20
WASHER	CF6-80	10
WASHER	CF6-80R	30

```
SELECT SUM(QTY)
FROM ENGINE
WHERE PART = 'BOLT'

SELECT SUM(QTY)
FROM ENGINE
WHERE PART = 'NUTS'

SELECT SUM(QTY)
FROM ENGINE
WHERE PART = 'SCREWS'

SELECT SUM(QTY)
FROM ENGINE
WHERE PART = 'WASHER'
```

ST_QTY
350
210
190
350

TOT_QTY
810

ENGINE STRUCTURE				
PART	CF6-50	CF6-80	CF6-80R	TOTAL
BOLT	90	120	140	350
NUTS	50	60	100	210
SCREWS	40	70	80	190
WASHER	20	10	30	60
TOTAL	200	260	350	810

Figure 1.5. Relational VS Multidimensional Data Consolidation.

For a more consistent response, system designers consolidate data and reload them into the database (Figure 1.5). In the relational format, precomputed totals speed up response. The relational process creates new summarized data using Standard Query Language (SQL). On the other hand, a multidimensional database consolidates data faster by simply adding up the row and column totals.

Both formats were developed for more efficient data delivery to users, but a multidimensional database complements data warehousing strategies. When a multidimensional database and a data warehouse work together, they enhance the quality, speed and efficiency of delivering corporate data to end users.

Dr. Kenan Sahin, president and CEO of Kenan Systems Corporation, defines the two functions of a multidimensional database when used as a data warehouse: (1) the

warehouse retailer model which provides user-friendly access and (2) the OLTP retailer model, which front-ends one or more OLTPs as well as open OLTP data to users.

To turn data into useful business information, a multidimensional database should have analysis tools, data integration and cleansing properties, as well as transparent data extraction from data warehouses. Many popular MDDB products in the market today share such properties: Arbor Software's Essbase, Comshare's Commander, Dimensional Insight's CrossTarget, Holistic Systems' Holos, Information Advantage's Axsys, Kenan Technologies' Accummate, MicroStrategy's DSS/Server, Oracle Personal Express, Pilot Software's LightShip Server, Planning Sciences' Gentium, Redbrick Systems' Redbrick Warehouse, Sniper's TM/1, and Standford Technology Group's Metacube.

A multidimensional database provides answers very fast. Aside from the fact that data are stored in coarser grains, another interesting aspect is that the information is stored in arrays, making updating possible without affecting the index. These features make multidimensional databases perfect for read-write applications.

1.4 Literature Reviews

In 1980, data warehousing began as a repository for data collection, then evolved into a business tool in information analysis. Today, due to a highly dynamic business environment, data warehousing is a decision-making tool because it offers fast, consistent and accurate information flow.

The data model is but one part in the data warehouse building process that creates the material views of management. [16]; [69]; [76]; [101]; [108]; [110]; [117]; [121]; [148]; [175]; [176]. A data structure is a graphic representation of data for a specific area of interest [171], which may be as broad as all data requirements of an organization, or as focused as a single business area or application. A data structure's

function is to convey clearly data, data relationships, data attributes, data definitions and the business rules that govern data.

Data models are the accepted way of representing and designing databases. Two data structuring approaches often used in a data warehousing are dimensional modeling (data structure), and ER modeling (conceptual structure) [17].

Data structuring is of two types: logical and physical [29]. A logical data structure is not a physical database but a graphic representation of the information requirements of a business area. It is pyramidal, enterprise-wide in scope and generic to all applications in the lower levels of the pyramid. It contains the identification and definition of all entities, relationships, and attributes. On the other hand, the physical data structure applies physical constraints, performance, and data distribution. This model is used to design actual physical implementation [150].

The two types of referring databases in the design of the data structures are the relational database [153]; [131] and the multidimensional database.

Various researches have defined the multidimensional data structure and the OLAP concept. [15]; [38]; [48]; [81]; [91]; [95]; [117]; [138]; [148]; [155]; [165]; [178]; [186]. These researches also review data structures in the form of applications, [1]; [10]; [11]; [102]; [112]; [127]; [130]; [167]; [185], analysis processes [28]; [49]; [63]; [129]; [136]; [164]; [187], and as other applied technical theories [2]; [30]; [34]; [106]; [114]; [133]; [140]; [168]; [177]; [192]. Past researches have also focussed on logical data structures [29]; [53], as well as on physical data structures [150]. Various researches on the physical data structure confirm the use of SQL language as query process in data loading, transformation, aggregation and summary [5]; [11]; [21]; [27]; [31]; [66]; [70]; [77]; [91]; [122]; [127]; [137]; [141]; [156]; [195].

Conceptual modeling is used to assist communication between analysts and end-users during data acquisition and specification verification. Modeling captures knowledge about the universe of discourse (uod) and represents it so as to enable a system developer to reason about this knowledge, communicate his/her understanding to end-users for verification and modify the model accordingly. The use of conceptual data structures must be guided by generic procedures so that developers may use the model's constructs in a standard approach within and across organizations.

Traditionally, conceptual models have paid more attention to the structural aspects of an application [3]; [14]; [33]; [38]; [74]; [177], giving rise to Semantic Data structures. The development of conceptual data structures has benefited from contributions from the fields of databases [10]; [127]; [167], artificial intelligence [119], programming languages [14]; [27]; [119], and software engineering [46]; [52]; [181].

In conceptual modeling, database, and case reference, there are three famous data structuring orientations: semantic data modeling, process modeling, and event modeling.

1.4.1. Semantic Data Modeling

Semantic data modeling approaches are concerned with the static aspects of an information system (i.e. objects, relationships, and integrity constraints). The basic approaches that dominate data modeling are: the ER diagram, graphic approaches, and binary-relationship approaches.

Various researches have focused on ER formalism [26]; [33]; [123]; [160], and on the binary-relationship approach [3]; [55]; [74]; [131]; [162]; [173]. Matteo Golfarelli, for instance, proposed a conceptual model for data warehouse design and a semi-automated methodology for deriving it from E/R documentation, which describes the

information system of the enterprise [17]. He also presents the design steps for E/R schemes for a typical health-care information system.

1.4.2. Process Modeling

Process modeling refers to data flow models, which are concerned with the specifications of activities in an application area of the uod. An activity is informally defined as a set of partially ordered sub-activities which themselves can be further decomposed. Activities are mainly concerned with modeling a uod in terms of flows of information in either direction [14]; [155].

1.4.3. Event Modeling

A major aspect of the relationship between time and information is that there are many such relationships. Two of these received the most attention: event time (real world time) at which a fact becomes valid, and transaction time (the time when this fact is recorded in the database). The most widely used definition of event modeling is "an instantaneous happening of interest to the enterprise" [26]; [46]; [52]; [119]; [123]; [160]; [181].

Most researches on data warehousing concepts focus on maintaining the warehouse with efficiency [59]; [73]; [82]; [83]; [98]; [100]; [85]; [84]; [124]; [145]; [146]; [147]; [189]; [190]; [191], and on warehouse performance consistency [94]; [196]; [197]; [198]; [199]; [200]. Nam Huyn, for instance, proposes a process of preserving views and keeping them consistent without updating the warehouse [85]; [84] by using an algorithm to generate SQL queries. The answers to the queries determine if a view can be maintained in a given situation. The answers also generate SQL updates that maintain the view. This is one process of view design that achieves a combination of good query performance and low view maintenance [94]. A framework is presented to highlight issues of materialized view design in a distributed data

warehouse environment. On the other hand, Yue Zhuge, et al presents three layers of consistency for materialized views in a distributed environment [196]; [199]; [200]. This consistency develops simple and scalable algorithms for achieving multiple views consistency (MVC) in a warehouse database.

The design of the data structure in multidimensional cubes is of primary importance prior to handling or maintaining the multidimensional views process. The lack of a data model will limit analysis to two-dimensional associations. On the other hand, an incomplete data model will result in error contamination in analytical procedures. The design of data structures have many other purposes [69]; [94]; [101]; [112]; [197]; [198], including algorithms and other methodologies to design either physical or logical data models of structures or metrics.

1.5 Data Structure Design Problems in Data Warehouse

Designing a hypercube database involves three major steps: First, the user decides on what particular business aspect to feature in the model. Second, the user identifies and assigns values, mostly numeric. Third, the user identifies data granularity, the lowest detail to be recorded. These three elements (business aspect, values, data granularity) are the three dimensions of a database constructed in hypercube.

The complicated design phases include: (1) preparing data from the warehouse, (2) designing the data structure for multidimensional views analysis, (3) developing the programs for loading data and other processes, (4) data distribution and data testing.

Designing a multidimensional database model requires the careful study of data in the existing operational system to prevent data incompatibility. Problems in the design phase include difficulty in handling large database, expertise required, and user requirement ambiguity.

1.5.1 Difficulty in Handling Large Databases

A data warehouse is a high-volume database, a repository fed by various sources with a data life-span of about five years. Transforming and distributing data results in an updated version of the warehouse database. The greatest activity involving the warehouse database is the extraction of data from relational databases into multidimensional databases.

Due to the huge amount of data inside a data warehouse, manual checking of data becomes a very complex task checking for errors is difficult. For example, the 'SEX' field is reserved only for 'M' or 'F' entries, but some entries are invalid. Manual checking of data in the voluminous warehouse is next to impossible. However, the automatic program proposed in this research will show all errors in a data warehouse in seconds.

1.5.2 Expertise Required

Data warehousing is a recent technology that requires a high degree of knowledge of various concepts of design and implementation of a modern and efficient data warehouse project. The manual design of a multidimensional data structure requires a high technical skills of human expertise. The data warehouse team is composed of Executive sponsors, Business Analysts, End-user Support, Technical Support, and MIS. Proper training, mentoring, planning, motivation, organization, and patience can help ensure that data warehouse team is position to provide the highest level of support, returning the highest return on investment to the organization. To solve the problems of a well designed data warehouse, the technical skills are required while the project time is limited. This research reduces the dependence on human expertise in designing a data structure by presenting a system that reviews data and automatically designs a multidimensional structure for the data.

1.5.3 User Requirement Ambiguity

The successful data warehouse project requires the involvement and the collaborative efforts of top executives, middle management, or professional leaders. The main problem with data warehousing is that most companies are creating data warehouse in a vacuum; identify and analyze key activities that are not essential to business missions, missing significant data items that served the user requirement, no building data models that support business factors of user requirements, etc. During process analysis and design, participants identify the specific data entities and attribute to support each process. Process is then mapped to legacy data sources and entities before the warehouse project moves forward to defining the physical data structure, this structure should initially built data model in support of the present and future business strategies of the organization.

Modest projects that may seem simple often are not. Managing user expectations and establishing acceptable compromises must be accomplished before expending a major portion of the budget and time. These projects must be carefully planned and the pattern of data presentation designed. Designing the data model and creating prototype may be a good idea if the user community understand what a data model proves and, more importantly, does not prove.

1.6 Scope and Objectives of the Research

The interconnected parts in Figure 1.1 are organized from various types of software tools and programming processes. The operational database and external database serve as the sources of up-to-date and historical data for analysts.

The Data Access Layer (DAL) allows the Information Access Layer to communicate with the Operational Database Layer. The DAL contents (i.e. data description, data type, source of data, etc.) are recorded into metadata or the Data

Directory Layer. The Data Staging Layer includes all process necessary to select, edit, summarize, combine and load data from operational and/or external databases. At this layer, there is complex programming in to create filters that identify patterns and data structures for meaningful data analysis. An increasing number of data warehousing tools are being created to help this process.

Finally, the Data Warehouse Layer is where the data is used by end users. In a physical data warehouse, many copies of operational and external data are actually stored in a form that is flexible and easy to access. Here, the volume of data increases all the time. To condense the data for easy archiving, it is rearranged in multidimensional view patterns. The process also cleans, transforms, and distributes the accumulating data from operational and/or external sources to the warehouse database. The complicated process is so delicate that only DSS analysts are allowed to handle them. They aggregate the interrelated layers into one, or sometimes distribute the processes to other responsible functions.

This research focuses on the part of the Data Access Layer that automatically generates a data structure to serve the specified multidimensional concept. The objectives and scope of this research are as follows:

1.6.1 Research Objectives

This research aims to provide two methods of creating a multidimensional data structure: dimensional modeling and graphs modeling. Instead of being individually designed by DSS analysts, both are automatically generated by computers to ensure a high-performance multidimensional database. This is needed especially in OLAP concepts.

The proposed research presents the three modules for automatic building of the logical DVR model of a warehouse database. A DVR model consists of a finite set of

grouping relationships. It is a combination of dimension attributes, variable attributes, and/or relative dimension attributes for decision-support requirements. The DVR model is used to create a hypercube that has the efficiency required for supporting OLAP applications.

This research also introduces the input-output design phases from the relational format into the multidimensional format using graph model. The research actually generates a logical multidimensional data structure in a graph presentation that is easily understood by end-users.

The proposed techniques in this research mainly focuses on 1) creating a logical multidimensional data structure to meet user requirements, 2) designing a routine that automatically creates a database format, 3) speeding up the multidimensional database input-output design phases, and 4) minimizing human errors and workload.

1.6.2 Research Scope

Related literature confirms that the two main modeling techniques for data warehousing processes today are dimension and E/R modeling, which can be used in relational or multidimensional formats.

This research delineates the steps to generate a DVR logical data model that is a multidimensional data structure, using MOLAP concepts. The scope of this research includes processes of creating logical data structures from current data level into summarized data level (Figure 1.2). This research also stresses three important components: dimensions, variables, and relative dimensions. All are vital relationships in the DVR model.

In the scope of building the data model concept, this research provides three modules for creating the DVR model: Classification, Partitioning, and Clustering Modules. These modules automatically create logical data structures from the relational

format in a warehouse database into the multidimensional format. In the scope of design phase using graph model, the designing of DVR data model is basically creating relationship of attributes in patterns of two-dimensional graph diagrams. Users are able to replace this pattern with a multidimensional graph diagram. This diagram will automatically generate the data structure in the same was as the prior concept. However, users can also modify the outcome of the logical data structure easily before transforming the data structure into a multidimensional database.

1.7 Organization of the Research

Chapter One presents background literature on data warehousing and architecture, database systems, multidimensional databases, and related research. It also presents the definition of the problem and the research scope and objectives.

Chapter Two presents the data structure and the On-Line Analytical Processing (OLAP) concept. The chapter also defines Dimensions, Variables, and Relative dimensions (DVR), vital in multidimensional database structuring.

Chapter Three presents the DVR model automatic generation procedure using three modules: classification, partitioning, and clustering. The DVR is a data structure in multidimensional hypercube.

Chapter Four presents a high-volume data processing procedure for transforming warehoused data into multidimensional form. To set up a data structure, a graph is designed to meet the particular needs of a specific group that will use the data.

Chapter Five presents other tasks to synchronize the various modules and summarizes the data structuring procedures for the automatic generation of a multidimensional database structure.

Data Transformation and data loading are in the appendix.

II. DATA STRUCTURE MODELING

2.1 Introduction

Big businesses accumulate so much data today that they become useless in speedy decision-making, even as the complexity of relationships between data increases the difficulty of analysis. Today, business needs a system that stores a great volume of data in multidimensional relationships and allows effective analysis of the relationships between them.

Data structure modeling refers to the creation of any of the various types of multidimensional database structures in order to meet specific data warehouse and end-user requirements.

Today, the process of changing relationship patterns from bidimensional to multidimensional continues to be done via manual interface. However, the increasing size and complexity of corporate databases as well as the need for faster analysis of highly complex patterns requires that the process be automated.

This research uses two DVR model-building concepts in a computer-generated program of data structure modeling that automatically transforms warehouse data into multidimensional DVR structures. The automatic process minimizes human errors. DVR structures are vital to fast and accurate decisions based on extremely large volumes of data.

This chapter discusses the following:

(1) Dimensions, Variables, and Relative dimensions (DVR)

(2) The DVR logical data structure

(3) Using OLAP tools to design data structures

(4) Designing the DVR data structure model

2.2 Approaches to Data Architecture

The DVR structure is a multidimensional database designed for storing and retrieving large volumes of data. Users access such data using the interactive OLAP tools. OLAP has many other applications, and one relevant to this research is its functions in creating data structure models.

The two most common data structuring techniques in data warehousing are ER (Entity Relationship) modeling and Dimensional Modeling [16]. ER modeling produces a specific data structure by using two basic concepts: entities and the relationships between those entities [109]; [110]; [112]; [115]. Detailed ER structures contain attributes, and properties of either entities or relationships. As an abstraction tool, the ER model is used to simplify and understand ambiguous data relationships in complex systems environments.

On the other hand, dimensional modeling uses measures, facts, and dimensions, and is effective in representing the requirements of user of database tables. Both ER and dimensional modeling can be used to create an abstract model of a specific subject, but each has a limited set of modeling approaches, concepts, and notation conventions. Thus, semantic representation techniques also differ.

2.2.1 Basic Entity-Relationship Modeling Concepts

An entity relationship (ER) structure is represented by an ER diagram, which uses three graphic symbols to represent the data: entity, relationship, and attribute (Figure 2.1).

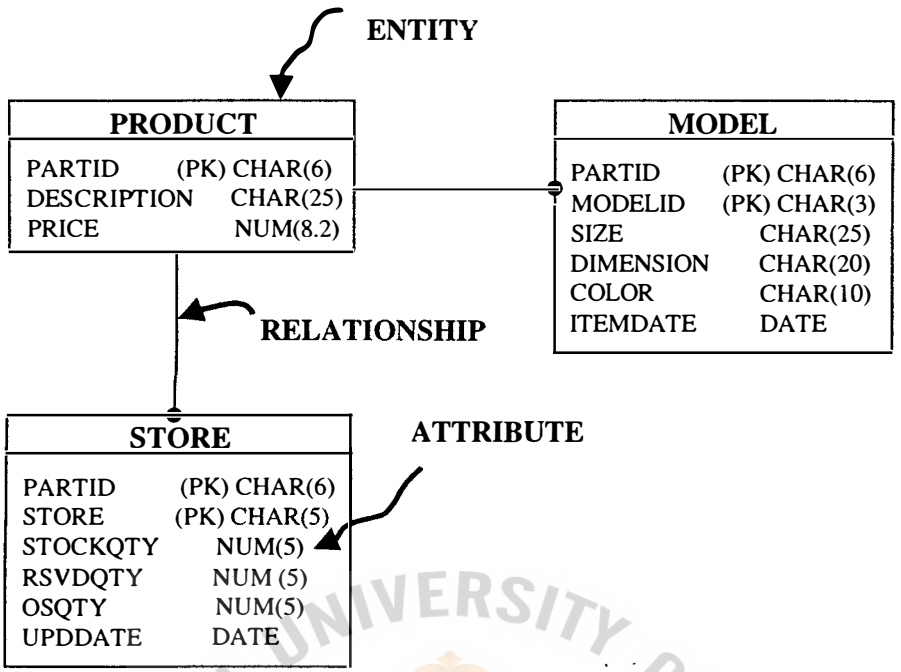


Figure 2.1. A Sample of ER Modeling.

(1) Entity

An entity is defined as a person, place, thing, or event, which may be relevant to the business or organization. An entity may also represent a class of tangible objects with their own properties and characteristics.

An entity usually has its own characteristics and limitations. In a practical modeling project, the project members share a single definition template for an integrated and consistent entity definition. In high-level business, entity modeling can be very generic but must be specific in the detailed logical structure.

In ER modeling, naming entities is essential, which is normally done by using nouns over verbs. An entity name should be a unique identifier and should represent the characteristics and scope of the entity very well. Called candidate keys, these unique identifiers form a set from which the primary (most common) key is selected.

(2) Relationship

Relationships are shown as lines between entities to depict the interaction and association among entities in a model or structure. A relationship is designated grammatically by a verb.

The relationship between two entities can be defined in terms of cardinality. This is the maximum number of instances that one entity can be related to a single instance in another table, and vice versa. The cardinalities may be: one-to-one (1:1), one-to-many (1:M), or many-to-many (M:M). In a detailed (normalized) ER model, M:M relationships are not shown; these are resolved to an associative entity.

(3) Attributes

Attributes are the characteristics or properties of the entities. Attribute naming conventions are very important. An attribute name should be unique and self-explanatory. In ER modeling, if the maximum cardinality of an attribute is more than 1, the modeler will try to normalize the entity and finally elevate the attribute to another entity. Therefore, the normal maximum cardinality of an attribute is 1.

2.2.2 Basic Dimensional Modeling Concepts

Dimensional modeling is a technique for conceptualizing and visualizing data models described by common elements. It is especially useful in data summary and rearrangement, as well as in data presentation for later analysis. Dimensional structures focus on numeric data, such as values, counts, weights, balances, and occurrences. Dimensional modeling has three basic concepts (Figure 2.2): facts, dimensions, and measures (variables).

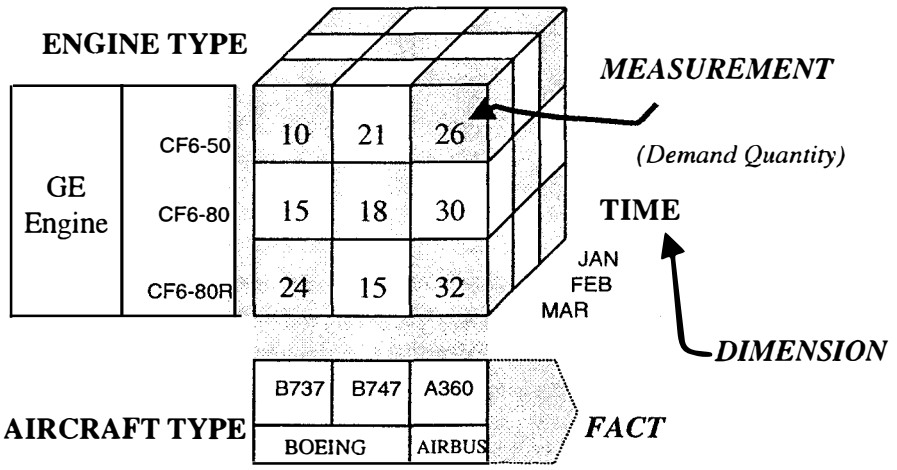


Figure 2.2. A Dimensional Data Structure.

(1) Fact

A fact is a collection of related data items, consisting of measures and context data. Each fact typically represents a business item, a business transaction, or an event that can be used in analyzing the business or business processes. In a data warehouse, facts are implemented in the core tables in which all of the numeric data is stored.

(2) Dimension

A dimension is a collection of members or units of the same type of views. In a diagram, a dimension is usually represented by an axis. In a dimensional model, every data point in the fact table is associated with one member from each of the multiple dimensions. That is, dimensions determine the contextual background for the facts. Many analytical processes are used to quantify the impact of dimensions on the facts. Dimensions are the parameters for Online Analytical Processing (OLAP). In a database analyzing product sales, for instance, the common dimensions could be time, location or region, customers, salespersons, or scenarios such as actual, budgeted, or estimated numbers.

Dimensions can usually be mapped to nonnumeric, informative entities such as branch or employee.

(a) Dimension Members: A dimension contains many dimension members. A dimension member is a distinct name or identifier used to determine a data item's position. For example, all months, quarters, and years make up a time dimension, while all cities, regions, and countries constitute a geography dimension.

(b) Dimension Hierarchies: The members of a dimension can be arranged into one or more hierarchies, each possibly having multiple hierarchy levels. This is because a dimension member may be present in several hierarchy structures. For example, the time dimension may have two hierarchies because a week can span two months, a quarter, and so on. Therefore, weeks cannot be added up to equal a month, for instance. If it is not practical to analyze the data on a weekly basis, then it is not necessary to assign another week hierarchy.

(3) Measurement

A measure is a numeric attribute of a fact, representing the performance or behavior of the business relative to the dimensions. The actual numbers are called variables, which may be expressed in terms of cash sales, sales volume, quantity supplied, supply cost, or transaction amount. A measure is determined by combinations of the members of the dimensions, and is located under facts.

Of the two most commonly used data models today, dimensional modeling is the simpler way of designing multidimensional views of OLAP applications. The discussion of DVR models in Chapter 3 includes the automatically-generated

dimensional structuring process, which transforms data from relational to multidimensional database using MOLAP tools.

On the other hand, Entity-Relationship modeling represents data structure models in graphical symbols related to ER diagrams. Some users may find it difficult to interpret ER diagrams. To make interpretation easier, the data structure is simply changed to one using a B-tree diagram.

The Chapter 4 discussion of logical DVRs using graph models explains the applications of ER modeling techniques in designing and translating multidimensional views into Graphical User Interface (GUI). Here, users can have a general view of the overall complexity of information inside a warehouse database. These techniques significantly minimize error contamination and human workload in designing database structures.

2.3 The DVR Model

The DVR structure model is a multidimensional database design for storing and retrieving large volumes of data. It is a finite set of grouping relationships that combine dimension attributes, variable attributes, and/or relative dimension attributes. This combination creates a structure of data storage, which is multidimensional in design. There are three basic components in a multidimensional data structure.

A dimension is a logical grouping of attributes with a common atomic key relationship [2]. The grouping is subject-oriented: i.e., product, location, and time.

A variable is fact or a measure that is normally stored as a numeric symbol, utilized to support investigation procedures [2].

The relative dimension or associated data description of a dimension is simply a relationship with an ordinary attribute of a group of data.

2.3.1 Dimensions

A dimension is a logical grouping of attributes with common atomic key relationships (Singh 1998). It is defined over a dimensional schema, a set of functionally interrelated dimensional attributes. The simple dimensional schema for customer dimension may consist of the attributes Customer, Location, and Country. The functional dependencies of customer dimension are presented as Customer → Location → Country.

Normally, dimensions are roughly equivalent to attributes in a relational database. For example, the attributes of engine structure shown in Figure 1.3. In a multidimensional database, "part" and "engine type" are dimensions; they are the key factors of business functions. A set of dimension grouping presents the relationships that interact to the variable (quantity attribute). The number at the interaction of each part and engine type occupies a cell, as in a spreadsheet, which is the result of combinations.

Each dimension has its own hierarchy that rolls up into only one total. In time dimension, the simple hierarchy is presented as monthly → quarterly → yearly. This means that each month belongs to only one quarter and each quarter belongs to only one year, and so on. Such concepts are useful to the end-user in that they allow the use of the hierarchies to "drill down" or "roll up" to successive levels of detail. OLAP applications today support these drill-down, roll-up functions.

Dimension has two main functions: as a subject-oriented grouping, and as decision-making elements.

As a subject-oriented grouping related to the variable dimensions can be any specific grouping of data, i.e., product, location, time.

The Order Number attribute is an example of dimension as a primary indexed key field, which is not a key factor in decision-making because number attributes have only the lowest level of granularity in the database.

As key factors in management decision-making, consider these questions of a manager:

(1) How many suppliers of shrimp are needed during the low tourist season?

(2) Which of the company's shrimp suppliers are in the top five rank?

(3) What month has the least passengers flying to New York City?

(4) Which top ten customers have over 10 million baht worth of orders this year?

(5) Which of these data has not changed over five-years?

The underlined words are the subjects for the decision-making process. Figure 2.3 shows three dimensions in a graph.

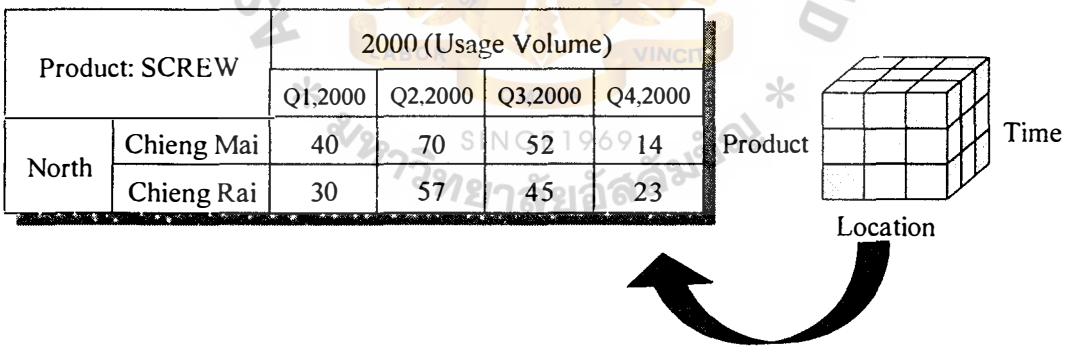


Figure 2.3. Data Dimensions in Graph Form.

2.3.2 Variables

Variables are numeric measures similar to value fields in a relational database (Figure 2.4). , "Cost," "Revenue," and "Expense," are examples of variables. A variable should correspond to specific dimensions in the database. For example, "Cost" might

be dimensioned by Time, Products, and Vendors. "Revenue," on the other hand, might be identical for all Products, Time, and Customers.

In designing a data structure model, a variable should be connected to the correct dimensions. This is the case in some OLAP products, where variables can have complex mathematical relationships to other variables. In this case, they are called complex variables.

A variable should be able to have very complex mathematical relationships between other variables. These relationships can include complex arithmetic operations, computed averages, time-lagged relationships, and even simultaneous equations.

When variables are summed up, they follow the rules of consolidation. For example, when costs are rolled up from Product to Total Product, the amounts are arithmetically added.

A derived variable appears to be a variable to the user but is actually computed on the fly at run time. For instance, Product Cost comes from calculations of product quantity and price attributes. It is a derived variable (also called measurement) because the value is computed using the formula (Cost = Quantity X Price). The measurement takes up no space in the multidimensional database because it will be stored only as a formula in metadata. This technique shrinks the size of a database and reduces consolidation time at the price of a small amount of overhead at run time performance.

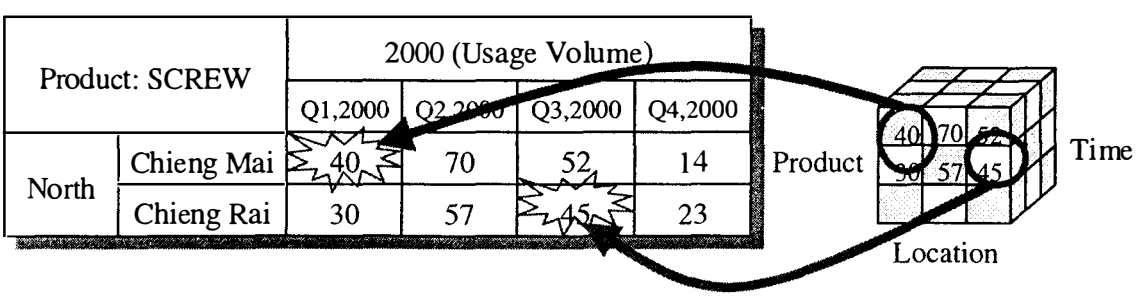


Figure 2.4. Multidimensional Variables on a Bidimensional Graph.

2.3.3 Relative Dimension

A relative dimension is the description variable, or text descriptions of dimension values (Figure 2.5). This is particularly useful when a dimension contains code values that are not meaningful to the user. Users prefer descriptive data over coded data.

A relative dimension is created the same way data variables are, and is normally stored in text format; some are in numeric form. However, a relative dimension relates to only one dimension while a variable relates to two or more dimensions.

An example of a relative dimension is the Product Name attribute that is directly associated with Product Code attribute.

PARTID	DESCRIPTION	PRICE	QTY
BLDE101	BLADESET	.....	.....
BLDE102	BLADESET-SIZE M	.....	.....
BLDE103	BLADESET-SIZE L	.....	.....
SCRW010	SCREW 1 x 2 INCH.	.....	.....
:	:	:	:
:	:	:	:

Figure 2.5. A Relative Dimension.

2.4 OLAP Methodology

In 1992, Codd first introduced OLAP, an acronym for On-Line Analytical Process. It is a software technology that allows fast, consistent, and interactive access to information so that people can make sense out of large volumes of data [113]; [114].

OLAP is often confused with decision support. In fact, OLAP reformulates relational data (flat files) into a multidimensional data store (hypercube). It stores data along specific dimensions that makes data easier to analyze and manipulate. OLAP uses

a wide range of views to transform raw data. This reveals other aspects of a business as understood by a user.

In sum, OLAP applications are characterized by the flexibility with which users can view and report data in any way they want; to perform new ad hoc analyses, to do large-scale complex calculations, and to perform dynamic reporting from large databases. A 1999 report of independent vendors and industry experts declared OLAP as a Fast Analysis of Shared Multidimensional Information (FASMI) [203].

Fast means the system delivers most responses to user requests within seconds, to minutes. This speed is hard to achieve in manipulating large volumes of data because a number could depend on millions of others. There are many techniques to solve this problem, including special forms of data storage, specific hardware requirements and extensive pre-calculations.

Analysis means the system copes with any business logic and statistical analysis that a user needs to use. A user is able to create new ad hoc calculations and to report on the data in any way without doing any programming. The various forms of analysis most relevant to business users include "slice and dice", "drill down", and "drill up."

Shared means that system implements all security requirements for confidentiality and concurrent access or locks at the appropriate levels.

Multidimensional means the system provides a multidimensional view of the data, including full support for hierarchies and multiple perspectives.

Information includes all data and derived data whenever these are needed or stored, and however much is required for an application.

OLAP and data warehousing are complementary; a data warehouse stores and manages physical data while OLAP transforms its data into strategic information. As decision-makers exercise more advanced OLAP capabilities, they move from data

access, to information, to knowledge. The key indicator of a successful OLAP application is its ability to provide information as needed, *i.e.*, its ability to provide "just-in-time" information for effective decision-making. This requires more than a base level of detailed data. OLAP systems have the ability to answer "who" and "what" questions, but what sets them apart from data warehouse is their ability to answer "what if" and "why" questions. OLAP also enables decision-making about future data via the following processes:

(1) Slicing and Dicing

Slice and dice processes enable end-users to cut or rotate a particular piece of data along any dimension. For example, "what product line generated the highest sales revenue in this country last year?"

(2) Drilling

The drill-down process allows users to navigate through information to get more detail and helps end-users answer "why" questions such as "Why did air tickets sales in the Asian region drop during the first and second quarters of 1998?" OLAP products allow access to various levels of detail within a dimension hierarchy. This is executed in many ways, including drill-down, drill-up, drill-across, and drill-around.

(3) Rotating

OLAP products allow a user to select one or more individual dimensions from one axis to the other. For example, one can flip the X and Y axes on a report with a click of a button.

(4) Ranking

Most OLAP allows a user to sort the output of a query from high to low or vice-versa.

(5) Paging

Paging allows a user to display a report in multiple page format. For example, a report can be made to show sales by product and by month, with a month's results on each page. The user tabs between pages to see each month's data. This function is particularly useful when viewing data in graph form because it is difficult to portray more than two dimensions on a single page of a graph.

(6) Filtering

A basic OLAP filter enables the user to limit the results for a query to a specific subset of the database. For example, a query can be as specific as this: "Show me sales by product, but only for the month of March".

Codd, et. al. [113] gave a 12-point criterion for evaluating OLAP products:

(1) Multidimensional Conceptual View. A user can view the records of a business multi-dimensionally and can manipulate such multidimensional data structure more easily and intuitively than a single-dimension data structure.

(2) Transparency. The OLAP application has an open system architecture that allows embedding anywhere the user desires.

(3) Accessibility. The OLAP application performs analysis based on a common conceptual schema of the business database.

(4) Consistent Reporting Performance. To maintain ease-of-use and simplicity the OLAP application has a consistent reporting performance even when dimensions or database volume increase.

(5) Client-Server Architecture. The OLAP applications or products operate in a client-server environment. OLAP tools in a server allows several users at the same time with minimal integration programming.

(6) Generic Dimensionality. Every data dimension is equivalent in both its structure and operational capabilities; the basic data structure, formulae, and reporting formats are not biased toward any data dimension.

(7) Dynamic Sparse Matrix Handling. The OLAP tools' physical schema adapt fully to the specific analytical model being created to provide optimal sparse matrix handling.

(8) Multi-user Support. The OLAP applications provide concurrent access, integrity, and security.

(9) Unrestricted Cross-Dimensional Operations. The OLAP tools infer the associated calculations and do not require users to define inherent calculations.

(10) Intuitive Data Manipulation. Manipulations (i.e., consolidate, drill down, zoom) are accomplished via direct action upon the cells of the analytical model, and should do not require menus or multiple trips across the user interface.

(11) Flexible Reporting. Analysis and presentation of data is simpler when rows, columns, and cells are arranged in proximity or by some logical grouping. Rows, columns, or page headings are each capable of containing or displaying from zero to n dimensions, where n is the number of dimensions in the entire analytical model.

(12) Unlimited Dimensions and Aggregation Levels. The OLAP applications accommodate unlimited numbers of data dimensions within a common analytical model.

Compared to OLTP (On-Line Transaction Processing), OLAP does more reading and aggregating high-volume data. It is different from OLTP applications, which does relatively simple transactions [2] such as retrieving and updating a small number of records in several tables simply related to each other. Examples of OLTP applications are inventory systems, human resource systems, bookkeeping, and payroll systems. The following table [2] compares OLTP and OLAP applications:

Table 2.1.The similarities and differences in OLTP and OLAP.

Description	OLTP	Data Warehouse (OLAP)
Purpose	Run daily operations	Information retrieval and analysis
Structure	RDBMS	RDBMS
Data Model	Normalized	Multidimensional
Access	SQL	SQL plus data analysis extensions
Type of Data	Data that runs the business	Data to analyze the business
Condition of Data	Changing, incomplete	Historical, descriptive

2.5 Conclusion

Because of the voluminous information that big corporations handle, decision making is more difficult today, so much so that there is a felt need for an automated, error-free storage system that allows effective and speedy analysis of the relationships between all that data. A DVR structure allows fast and accurate decisions based on extremely large volumes of data.

This chapter presents the DVR model and how it is constructed using OLAP. It discusses how data relationships are changed using a computer-generated program that creates any desired data structure as well as automatically transforms warehouse data into multidimensional DVR structures (hypercubes).

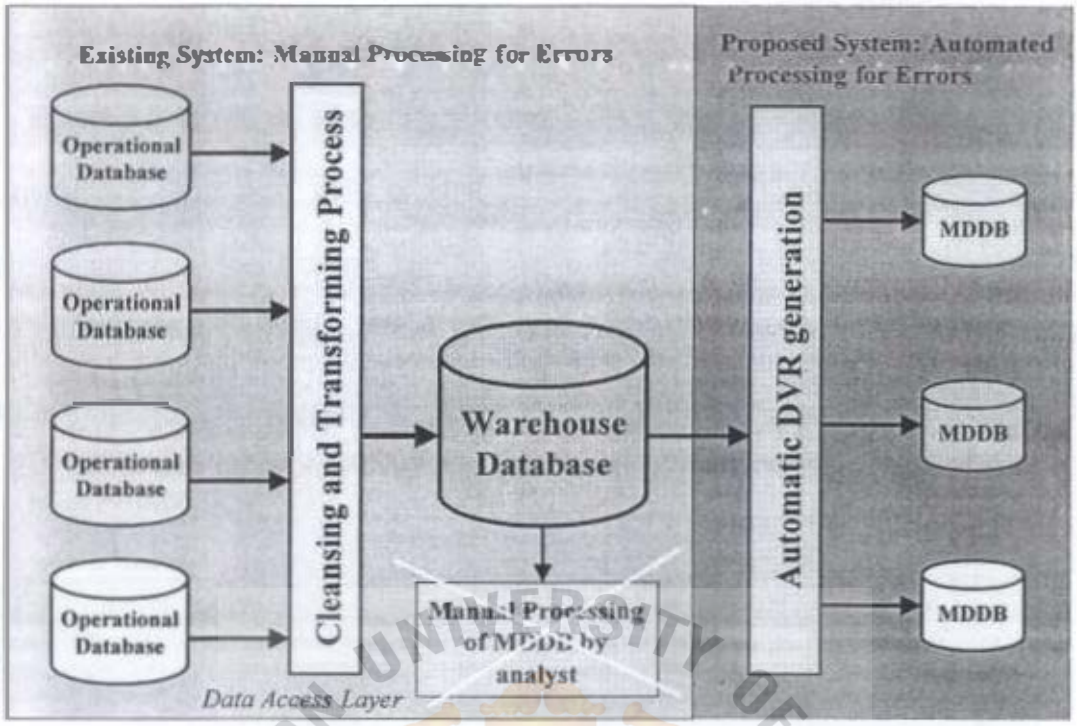
III. DVR DATA MODEL GENERATION

3.1 Introduction

This chapter discusses how to generate multidimensional DVR data structures using fast and efficient data mining technology and procedures. DVR is a program for storing data in multidimensional hypercube. The acronym means Dimensions, Variables, and Relative dimensions.

Warehoused data is a large collection of information in a computer. To compress, store, access, and analyze these data in relation to other data in the warehouse, the author stored data in a structure that allows very fluid manipulation. This structure is called hypercube, a multidimensional format. However, changing flat data in a warehouse into hypercube is a time-consuming routine that invites human errors. This research presents the process of creating data model that does the job faster, and with less human input. This chapter explains the automatic procedure for generating a DVR data structure using three data mining modules: classification, partitioning, and clustering. The process yields faster analysis of more complex patterns, as well as screens out data that are not relevant to the analysis.

Without a DVR data structure in multidimensional cubes, analysis is limited to two-dimensional associations, while an incomplete DVR model causes errors in analysis. An MDDB or multidimensional database effectively avoids such problems.



PARADIGM OF THE RESEARCH

Figure 3.1. The Research Paradigm.

Figure 3.1, the paradigm of the research, shows the processes of transforming flat warehoused data into a multidimensional database. First, various operational databases feed data into a warehouse database and store the data in relational format. Second, at the data access layer, various tools or programs transform or screen the data. At this point, data is voluminous because of low-level granularity. To analyze the database for errors, manual processing is needed. This research proposes an automatic process for generating DVR, which summarizes data into MDDBs. This process is faster and less error-prone, but does not allow manual processing for errors. However, corrections to the original data or the procedure may be done manually at the two interface nodes (Figure 3.2).

3.2 Generating the DVR Data Structure

Figure 3.2 shows the steps in generating a DVR data structure from data stored in bidimensional form into data in multidimensional form. Data in the warehouse databases undergo classification, partitioning, and clustering to be transformed into the DVR structure. The transformed data is then relocated and loaded. Two interface nodes in the automated generating process allows for manual data or procedural corrections.

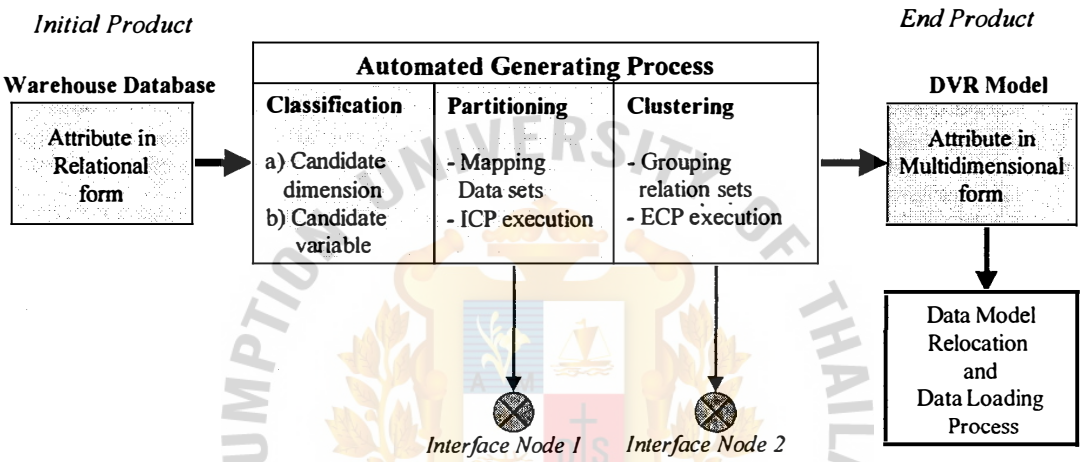


Figure 3.2. DVR Data Structure Generating Process.

(1) The Classification Module. This module inspects all data types for indexed keys and verifies the data entity and the candidate dimension of multidimensional cubes. If the attribute is verified as numeric, the data is classified as variables.

(2) The Partitioning Module. To form mapping data sets, the candidate dimensions from the classification module are used as bases for mapping with the other attributes of the same data entity. The data sets are manipulated using the Internal Comparison Process (ICP), in which each attribute in the data set is assigned as actual dimension, variable or relative dimension. The module produces a DVR data model of each data entity.

(3) The Clustering Module. The clustering module uses data sets of particular entities as bases for mapping with other data sets of various data entities. To generate a new data set that shows the relationship between the base data and the rest of the data sets, the mapped data sets are processed using the External Comparison Process (ECP). This module generates new DVR data model of multidimensional cubes using a union function of DVR data models from various data entities.

The resulting DVR model is analyzed for suitability to user needs and satisfaction. This guides the consequent refining process. Inaccuracies in the original data as well as corrections in the procedure may be done manually at the two interface nodes. Finally, the generated DVR data structure guides the creation of new DVR data structures.

3.2.1 Classification Module

The classification module first indicates the candidate dimensions (variables) of each attribute in the data entity. Then, to establish the DVR indication, it compares the attribute with the definite rule of the DVR model. (This module is currently limited to classifying candidate dimension and variable.) For simplicity of notation, this research denotes the candidate dimension as D_x^i and the candidate variable as V_x^i, corresponding to an attribute x at data entity, or data entity i in the research respectively.

(1) Candidate Dimension Selection

A dimension is a group of attributes with a common relationship [172]. The grouping is subject-oriented: i.e., product, location and time. When an attribute in a data entity is established for creating a DVR model, its data type is classified as candidate dimension and recorded in the data definition. Any candidate dimension is reclassified as actual dimension in the partitioning module. The propositional function, a formal way of

representing knowledge in terms of declarative sentences [97], shows this procedure:

For any attribute x of a data entity, if attribute x is an index key field or its data is either in character or date format, attribute x is defined as *candidate dimension*. Thus,

$$(\forall x)\left(I_x \cup \left(\delta_{cx} \cup \delta_{tx}\right)\right) \Rightarrow D_x^i$$

Where, $\boldsymbol{I_x}$ is the index key format, normally stored as key field,

δ_x is the type of data property is equivalent to character format (c), or date/time format (t), and

D_x^i is the attribute x within dimension of data entity i, when $i > 0$.

(2) Candidate Variable Selection

A variable is a fact or a measure normally stored as a numeric field. This has been the focus of a decision-support investigation [172]. After the candidate dimensions are indicated, the remaining attributes dictate the selection of candidate variables. The propositional function expresses this process:

For any attribute x, if numeric (binary, integer, or decimal), define attribute x as "candidate variable," thus:

$$(\forall x)\left(\delta_{nx}\right) \Rightarrow V_x^i$$

where δn is data type equals to numeric format (denoted by "n"), and

V_x^i is attribute x within the variable of data entity i, when i > 0.

The term relative dimension, or associated data description of a dimension, is simply a relationship with an ordinary attribute of a grouping relation. Initially, it is too

complicated to be classified by the argument that the alternative is to convert it into a candidate dimension.

The last step of the partitioning module repeatedly predicates the characteristic of actual and relative dimensions.

The following subroutines present a sample criterion for selecting candidate dimension and candidate variable using Basic Programming. These subroutines select the database, identify the data type of each attribute within a data entity, then display the output into a listbox. The database is from Microsoft Access 97, the programming language is Visual Basic (Microsoft, Version 5.0).

Example 1: The data entity *SALE* and data entity COST in a warehouse database is shown as relational tables (see Table 3.1, Table 3.2):

Table 3.1. SALE Data Entity.

Attribute Name	Data Type	Length
SCODE (Saleman code) [Index key]	CHAR	5
SNAME (Saleman Name)	CHAR	30
ACODE (Area code) [Foreign key]	CHAR	3
SALEVAL (Sale value)	NUM	10.2
COMM (Commission)	NUM	10.2

Table 3.2. COST Data Entity.

Attribute Name	Data Type	Length
CCODE (Customer code) [Index key]	CHAR	6
PCODE (Product Code) [Index key]	CHAR	5
PNAME (Product Name)	CHAR	30
SCODE (Saleman code) [Foreign key]	CHAR	5
SALEVAL (Sale value)	NUM	10.2
COSTVAL (Cost value)	NUM	10.2

When the fundamental characteristics of the attributes *SCODE, SNAME, ACODE* are presented in the index-key field and/or character formats of data entity SALE, these

characteristics are defined as candidate dimension. Since the data types of attributes *SALEVAL* and *COMM* are in the numeric format, these characteristics are classified as candidate variable.

The COST data entity's attributes of *CCODE, PCODE, PNAME,* and *SCODE* are in the index-key field and/or character formats and thus defined as candidate dimension. On the other hand, the data types of attributes *SALEVAL* and *COSTVAL* are numeric and are defined as candidate variable.

3.2.2 The Partitioning Module

In the classification module, all attributes of data entities are categorized as candidate dimension and candidate variable, relevant to the grouping of associated attributes in the partitioning module. The partitioning module process includes mapping, and internal comparison process (ICP).

The mapping process assigns candidate dimensions as base attributes and maps these with the same data entity attributes for mapping data sets. The mapping data sets are then manipulated with the ICP to achieve the actual dimension, variable, and relative dimension. The following illustrates the entire process of mapping data sets.

Consider a warehouse database denoted as DB_i. Assume the data entities in DB_i are *A, B, C:* (DB_i = *A, B, C)*; Suppose $A = \{a_1, a_2, a_3, a_4\}$, $B = \{a_1, a_5, a_6\}$ and $C = \{a_5, a_7\}$ where *a* is an attribute. Assume a_1 and a_2 in data entity *A* are candidate dimensions, while a_3 and a_4 are candidate variables. Attributes a_1 and a_2 are mapped together with other attributes in data entity *A*. Let $\overline{A}$ be the mapped data sets of *A*.

This concept is presented thus:

$$\overline{A} = \begin{cases} (a_1 \infty a_2), (a_1 \infty a_3), (a_1 \infty a_4), (a_1 \infty a_2 \infty a_3), (a_1 \infty a_2 \infty a_4), (a_1 \infty a_3 \infty a_4), (a_1 \infty a_2 \infty a_3 \infty a_4), \\ (a_2 \infty a_1), (a_2 \infty a_3), (a_2 \infty a_4), (a_2 \infty a_1 \infty a_3), (a_2 \infty a_1 \infty a_4), (a_2 \infty a_3 \infty a_4), (a_2 \infty a_1 \infty a_3 \infty a_4) \end{cases}$$

To prove the mapping of data set $\overline{A}$, base attributes and consequence attributes are substituted into a mapping table as shown below (see Table 3.3).

Table 3.3. Mapping table of partitioning module.

Base Attributes	Attributes of A				Mapping Data Set
	a_1	a_2	a_3	a_4	
a_1	-	X	-	-	$a_1 \infty a_2$
a_1	-	-	X	-	$a_1 \infty a_3$
a_1	-	-	-	X	$a_1 \infty a_4$
a_1	-	X	X	-	$a_1 \infty a_2 \infty a_3$
a_1	-	X	-	X	$a_1 \infty a_2 \infty a_4$
a_1	-	-	X	X	$a_1 \infty a_3 \infty a_4$
a_1	-	X	X	X	$a_1 \infty a_2 \infty a_3 \infty a_4$
a_2	X	-	-	-	$a_2 \infty a_1$
a_2	-	-	X	-	$a_2 \infty a_3$
a_2	-	-	-	X	$a_2 \infty a_4$
a_2	X	-	X	-	$a_2 \infty a_1 \infty a_3$
a_2	X	-	-	X	$a_2 \infty a_1 \infty a_4$
a_2	-	-	X	X	$a_2 \infty a_3 \infty a_4$
a_2	-	-	X	X	$a_2 \infty a_1 \infty a_3 \infty a_4$

The mapping data sets are then manipulated using ICP, in which each data set is validated to form the DVR model. The ICP processes follows.

PROCESS 1: According to the commutativity of classical set operations, when mapping data sets are symmetric and represent similar attributes, a data set is invited to form a DVR model.

Process 1 follows the basic concepts of classical set theory. Assuming data sets $(a_1 \infty a_2)$ and $(a_2 \infty a_1)$ are accomplished, this proves that the attributes of both data sets are similar and they belong to equal sets. Accordingly a data set is accessed in uniqueness. The execution of process 1 is:

$$P_1 = \begin{cases} (a_1 \infty a_2), (a_1 \infty a_3), (a_1 \infty a_4), (a_1 \infty a_2 \infty a_3), (a_1 \infty a_2 \infty a_4), (a_1 \infty a_3 \infty a_4), \\ (a_2 \infty a_3), (a_2 \infty a_4), (a_2 \infty a_3 \infty a_4), (a_1 \infty a_2 \infty a_3 \infty a_4) \end{cases}$$

PROCESS 2: The base attribute of a data set must become a candidate dimension, otherwise the data set is disregarded.

The first attribute of a data set in Process 2 should only be a candidate dimension. Consequent attributes may be either candidate dimension or candidate variable. Process 2 does not allow the replacement of a first attribute with a candidate variable because the variable is related to at least one dimension. The execution of process 2 is:

$$P_2 = \begin{cases} (a_1 \infty a_2), (a_1 \infty a_3), (a_1 \infty a_4), (a_1 \infty a_2 \infty a_3), (a_1 \infty a_2 \infty a_4), (a_1 \infty a_3 \infty a_4), \\ (a_2 \infty a_3), (a_2 \infty a_4), (a_2 \infty a_3 \infty a_4), (a_1 \infty a_2 \infty a_3 \infty a_4) \end{cases}$$

PROCESS 3: After completing Process 1, if the attributes in the data set are candidate dimensions, check each of the characters using the SQL process to count the value of each attribute and compare them. If the values of two attributes are similar, proceed to Process 4. Otherwise, split each attribute into independent dimensions.

Assuming data set $(a_1 \infty a_2)$ is achieved, attributes a_1 and a_2 sift the volume of data items using the SQL structure (select-joint predicates) in order to compare their relationships. The simple comparison of data set $(a_1 \infty a_2)$ is predicated by SQL function that is presented as $\Gamma^{a_x}_{count}(DIST(a_x)) = \Gamma^{a_y}_{count}(DIST(a_y))$, where the symbol $\Gamma_m{}^n$ means *query process* in which m represents the SQL aggregate functions (*i.e.* count, sum, max, min) and n is grouped according to attribute.

The SQL computation shows that aggregated data value a_1 is x and aggregated data value a_2 is y. When the aggregated data value x is equivalent to data value y, this proves a relationship between attribute a_1 and a_2 in the data entity. Thus, a data set is predicated as a grouping relation set that is composed of actual and relative dimensions.

On the other hand, when the aggregated data value x is dissimilar to y, and attributes a_1 and a_2 are predicated as candidate dimensions, the program divides them into new actual dimensions.

PROCESS 4: Since the data set of Process 3 may result in two attributes that are identified to be actual dimension and relative dimension. Examine the data set to find the attribute present in the index key and classify that as actual dimension. Classify the other as relative dimension. Replace the joint function symbol (∞) with the related function symbol (↔).

Process 4 indicates the data property of attributes in a data set. Assuming that data set $(a_1 \infty a_2)$ is the grouping relation set and defined as candidate dimension in the classification module, when the data type of attribute a_1 is an index-key field (primary or foreign key) and the data type of attribute a_2 isn't, attribute a_1 is defined as actual dimension (D_x). Attribute a_2 is defined as relative dimension (R_x) of actual dimension (D_x). The data set $(a_1 \infty a_2)$ becomes $(a_1 \leftrightarrow a_2)$. If the attributes are index-key fields, they are divided into new actual dimensions. Assuming a_2 is a relative dimension, the execution of process 4 is:

$$P_4 = \begin{cases} (a_1 \leftrightarrow a_2), (a_1 \infty a_3), (a_1 \infty a_4), (a_1 \infty a_2 \infty a_3), (a_1 \infty a_2 \infty a_4), (a_1 \infty a_3 \infty a_4), \\ (a_2 \infty a_3), (a_2 \infty a_4), (a_2 \infty a_3 \infty a_4), (a_1 \infty a_2 \infty a_3 \infty a_4) \end{cases}$$

PROCESS 5: Process 4 yields a data set that presents actual and relative dimensions. Other data sets with the same attributes as the relative dimensions will be deleted.

For example, when data set $(a_1 \leftrightarrow a_2)$ is processed, all remaining data sets composed of attribute a_2, such as data sets $(a_1 \infty a_2 \infty a_3)$ $(a_1 \infty a_2 \infty a_4)$, are erased. The execution of process 5 is:

$$P_5 = \{(a_1 \leftrightarrow a_2), (a_1 \infty a_3), (a_1 \infty a_4), (a_1 \infty a_3 \infty a_4)\}$$

PROCESS 6: When the consequent attributes are indicated as candidate variables, the mapping data set is superseded and the joint function (∞) is replaced with the associated function (⇒).

In the example of data set $(a_1 \infty a_3)$, when a_3 is predicated as candidate variable, it replaces the joint function symbol with the associated function symbol: $(a_1 \Rightarrow a_3)$. The execution of process 6 is:

$$P_6 = \{(a_1 \leftrightarrow a_2),(a_1 \Rightarrow a_3),(a_1 \Rightarrow a_4),(a_1 \Rightarrow a_3, a_4)\}$$

PROCESS 7: To prevent duplication of data sets, whenever the homogeneous base attributes indicate a dimension and consequent attributes are indicated as variables in any data set, only one completed data set is retained. The rest are deleted.

The various data sets such as $(a_1 \Rightarrow a_3)$, $(a_1 \Rightarrow a_4)$ and $(a_1 \Rightarrow a_3, a_4)$ in which base attributes are similar (attribute a_1) and the latter attributes are variables, then $(a_1 \Rightarrow a_3, a_4)$ is the completed data set that includes variables a_3 and a_4. To prevent data set duplication, data sets $(a_1 \Rightarrow a_3)$ and $(a_1 \Rightarrow a_4)$ are deleted. The execution of Process 7 is:

$$P_7 = \{(a_1 \leftrightarrow a_2),(a_1 \Rightarrow a_3, a_4)\}$$

PROCESS 8: Attributes represented as dimensions are replaced by D_x, where x is the dimension number. The sign V_x replaces the attributes that are represented as variable, where x is the variable number. Finally, the sign R_x replaces the attributes that are represented as relative dimension, where x is the relative dimension number. The execution of Process 8 is:

$$P_8 = D_1.a_1 \leftrightarrow R_1.a_2, D_1.a_1 \Rightarrow V_1.a_3, V_2.a_4$$

The following example illustrates the ICP steps of building a logical DVR model:

Example 2.1: Assume data entity 1 of warehouse database is *SALE* data entity consisting of attributes *SCODE* (Saleman Code), *SNAME* (Saleman Name), *ACODE* (Area Code), *SALEVAL* (Sale Value), and *COMM* (Commission Value) shown in Table 3.4.

Table 3.4. SALE Data Entity and DVR Indicator.

Attribute Name	Data Type	Length	DVR Forms
1. SCODE (Index Key)	Char	5	D
2. SNAME	Char	30	D
3. ACODE (Foreign Key)	Char	3	D
4. SALEVAL	Num	10.2	V
5. COMM	Num	10.2	V

First, let attribute *SCODE* be a_1, attribute *SNAME* be a_2, attribute *ACODE* be a_3, attribute *SALEVAL* be a_4, and attribute *COMM* be a_5. Each candidate dimension (eg. a_1, a_2, a_3) is used as base attributes and performs mapping with other attributes of *SALE* data entity.

The result of mapping data sets is:

$$\overline{A} = \overline{A_1} \cup \overline{A_2} \cup \overline{A_3}$$

$$\overline{A_1} = \{(a_1 \infty a_2),(a_1 \infty a_3),(a_1 \infty a_4),(a_1 \infty a_5),(a_1 \infty a_2 \infty a_3),(a_1 \infty a_2 \infty a_4),(a_1 \infty a_2 \infty a_5),(a_1 \infty a_3 \infty a_4),(a_1 \infty a_3 \infty a_5),(a_1 \infty a_4 \infty a_5),(a_1 \infty a_2 \infty a_3 \infty a_4),(a_1 \infty a_2 \infty a_3 \infty a_5),(a_1 \infty a_3 \infty a_4 \infty a_5),(a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5)\}$$

$$\overline{A_2} = \{(a_2 \infty a_1),(a_2 \infty a_3),(a_2 \infty a_4),(a_2 \infty a_5),(a_2 \infty a_1 \infty a_3),(a_2 \infty a_1 \infty a_4),(a_2 \infty a_1 \infty a_5),(a_2 \infty a_3 \infty a_4),(a_2 \infty a_3 \infty a_5),(a_2 \infty a_4 \infty a_5),(a_2 \infty a_1 \infty a_3 \infty a_4),(a_2 \infty a_1 \infty a_3 \infty a_5),(a_2 \infty a_3 \infty a_4 \infty a_5),(a_2 \infty a_1 \infty a_3 \infty a_4 \infty a_5)\}$$

$$\overline{A_3} = \{(a_3 \infty a_1),(a_3 \infty a_2),(a_3 \infty a_4),(a_3 \infty a_5),(a_3 \infty a_1 \infty a_2),(a_3 \infty a_1 \infty a_4),(a_3 \infty a_1 \infty a_5),(a_3 \infty a_2 \infty a_4),(a_3 \infty a_2 \infty a_5),(a_3 \infty a_4 \infty a_5),(a_3 \infty a_1 \infty a_2 \infty a_4),(a_3 \infty a_1 \infty a_2 \infty a_5),(a_3 \infty a_2 \infty a_4 \infty a_5),(a_3 \infty a_1 \infty a_2 \infty a_4 \infty a_5)\}$$

Assume that the sample data elements are as shown in Table 3.5 below.

Table 3.5. Data Element of SALE Data Entity.

Rec.	SCODE	SNAME	ACODE	SALEVAL	COMM
1	*10200*	*John Smith*	*EAST*	*$200,000.00*	*$1,000.00*
2	*10200*	*John Smith*	*WEST*	*$150,000.00*	*$750.00*
3	*10200*	*John Smith*	*NORTH*	*$750,000.00*	*$3,750.00*
4	*10210*	*Caroline Jone*	*SOUTH*	*$59,000.00*	*$295.00*
5	*10210*	*Caroline Jone*	*WEST*	*$650,000.00*	*$3,250.00*
6	*10240*	*Tom Hogan*	*WEST*	*$410,000.00*	*$2,050.00*

The result of $\overline{A}$ is prepared for access in ICP in order to validate the DVR structure. In the case of duplicate data sets, Process 1 selects one and deletes the rest. When data sets ($a_1 \infty a_2$) and ($a_2 \infty a_1$) are found similar, one data set is selected, the rest deleted:

$$\overline{A} = \begin{cases} (a_1 \infty a_2), (a_1 \infty a_3), (a_1 \infty a_4), (a_1 \infty a_5), (a_1 \infty a_2 \infty a_3), (a_1 \infty a_2 \infty a_4), (a_1 \infty a_2 \infty a_5), \\ (a_1 \infty a_3 \infty a_4), (a_1 \infty a_3 \infty a_5), (a_1 \infty a_4 \infty a_5), (a_1 \infty a_2 \infty a_3 \infty a_4), (a_1 \infty a_2 \infty a_3 \infty a_5), \\ (a_1 \infty a_3 \infty a_4 \infty a_5), (a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5), (a_2 \infty a_3), (a_2 \infty a_4), (a_2 \infty a_5), (a_2 \infty a_3 \infty a_4), \\ (a_2 \infty a_3 \infty a_5), (a_2 \infty a_4 \infty a_5), (a_2 \infty a_3 \infty a_4 \infty a_5), (a_3 \infty a_4), (a_3 \infty a_5), (a_3 \infty a_4 \infty a_5) \end{cases}$$

The $\overline{A}$ data sets undergo Process 3. In the example of data set ($a_1 \infty a_2$), both are verified as candidate dimensions by the classification module. The comparison of attributes a_1 and a_2 in data set ($a_1 \infty a_2$) use this query process:

```
SELECT count(*)                    SELECT count(*)
FROM SALE                          FROM SALE
GROUP BY SCODE                     GROUP BY SNAME
ORDER BY SCODE                     ORDER BY SNAME
```

The values of $\Gamma^{a_1}_{count}\left(DIST(a_1)\right) = 3$ and $\Gamma^{a_2}_{count}\left(DIST(a_2)\right) = 3$, show that the attributes a_1 and a_2 are related. Therefore, this data set accesses the next algorithm. If the solution of attributes a_1 and a_3 of data set ($a_1 \infty a_3$) is dissimilar, the process automatically represents them as *dimensions*: attribute a_1 as D_1 ($D_1.SCODE$) and attribute a_3 as D_2 ($D_2.ACODE$).

In data set ($a_1 \infty a_2$), when the data type of attribute a_1 is an index-key field, the process automatically represents attribute a_1 as a dimension ($D_1.SCODE$) and attribute

a_2 as a relative dimension ($R_1.SNAME$). The data set is filled in for new data set ($a_1 \leftrightarrow a_2$). In Process 2, when attribute a_2 is represented as relation, then the mapping of data sets in which attribute a_2 is used as base is cancelled.

In Processes 5 and 6, the existing data sets composed of relative dimensions (attribute a_2) are erased. An example is data sets ($a_1 \infty a_2 \infty a_3$), ($a_1 \infty a_2 \infty a_4$), ($a_1 \infty a_2 \infty a_5$), ($a_1 \infty a_2 \infty a_3 \infty a_4$), ($a_1 \infty a_2 \infty a_3 \infty a_5$), ($a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5$), ($a_2 \infty a_3 \infty a_5$), ($a_2 \infty a_4 \infty a_5$), ($a_2 \infty a_3 \infty a_4 \infty a_5$). When the remaining data sets that are composed of dimensions and variables are replaced by the symbol ($\Rightarrow$), the results are:

$$\overline{A} = \begin{cases} (a_1 \leftrightarrow a_2), (a_1 \infty a_3), (a_1 \Rightarrow a_4), (a_1 \Rightarrow a_5), \\ (a_1 \infty a_3 \Rightarrow a_4), (a_1 \infty a_3 \Rightarrow a_5), (a_1 \Rightarrow a_4, a_5), (a_1 \infty a_3 \Rightarrow a_4, a_5), \\ (a_3 \Rightarrow a_4), (a_3 \Rightarrow a_5), (a_3 \Rightarrow a_4, a_5) \end{cases}$$

The different data values of any dimension in mapping data sets are integrated into unique data sets in Process 7. For example, data sets ($a_1 \Rightarrow a_4$) and ($a_1 \Rightarrow a_5$) are integrated as ($a_1 \Rightarrow a_4, a_5$) and thus become:

$$\overline{A} = \{(a_1 \leftrightarrow a_2), (a_1 \Rightarrow a_4, a_5), (a_1 \infty a_3 \Rightarrow a_4, a_5), (a_3 \Rightarrow a_4, a_5)$$

As attributes a_1 and a_3 are used as dimensions in any steps of *ICP*, they are represented as actual dimensions. The correspondence of the *SQL* statement of the data set ($a_1 \Rightarrow a_4, a_5$) is presented as:

```
SELECT distinct SCODE, sum(SALEVAL),sum(COMM)
FROM SALE
ORDER BY SCODE
GROUP BY SCODE
```

Where attribute a_1 is replaced with *SCODE*, attribute a_4 is replaced with *SALEVAL*, and attribute a_5 is replaced with *COMM*, the elements of dimension $D_1.SCODE$ and associated data values $V_1.SALEVAL$, $V_2.COMM$ are presented as Table 3.6:

Table 3.6. Sample Data of SCODE Dimension.

Rec.	SCODE	SALEVAL	COMM
1	*10200*	*$1,100,000.00*	*$5,500.00*
2	*10210*	*$709,000.00*	*$3,545.00*
3	*10240*	*$410,000.00*	*$2,050.00*

The results of the data sets are replaced with the sign of a *DVR* structure from Process 8, represented as: $D_1.SCODE$, $R_1.SNAME$, $D_2..ACODE$, $V_1.COMM$. The following are *DVR* models of this example (see Table 3.7, 3.8, 3.9, 3.10):

Table 3.7. $D_1.SCODE \Rightarrow V_1.SALEVAL$, $V_2.COMM$.

SCODE	SALEVAL	COMM
10200	*$1,100,000*	*$5,500*
10210	*$709,000*	*$3,545*
10240	*$410,000*	*$2,050*

Table 3.8. $D_1.SCODE \leftrightarrow R_1.SNAME$.

SCODE	SNAME
10200	*John Smith*
10210	*Caroline Jone*
10240	*Tom Hogan*

Table 3.9. $D_2.ACODE \Rightarrow V_1.SALEVAL$, $V_2.COMM$.

ACODE	SALEVAL	COMM
EAST	*$200,000*	*$1,000*
NORTH	*$750,000*	*$3,750*
SOUTH	*$59,000*	*$295*
WEST	*$1,210,000*	*$6,050*

Table 3.10. $D_1.SCODE \infty D_2.ACODE \Rightarrow V_1.SALEVAL, V_2.COMM.$

SCODE	ACODE	SALEVAL	COMM
10200	*EAST*	*$200,000*	*$1,000*
10200	*WEST*	*$150,000*	*$750*
10200	*NORTH*	*$750,000*	*$3,750*
10210	*SOUTH*	*$59,000*	*$295*
10210	*WEST*	*$650,000*	*$3,250*
10240	WEST	$410,000	$2,050

The above samples are data values actually generated from *DVR* models. Data analysts may modify *DVR* structures manually at any of the two interface nodes (*Figure 3.2*).

Example 2.2: Assume data entity 2 of warehouse database is *COST* and consists of attributes *CCODE* (Customer Code), *PCODE* (Product Code), *PNAME* (Product Name), *SCODE* (Saleman Code), *SALEVAL* (Sale Value), and *COSTVAL* (Cost Value) as shown in Table 3.11.

Table 3.11. COST Data Entity and DVR Indicator.

Attribute Name	Data Type	Length	DVR Forms
CCODE (Customer code)	CHAR	6	D
PCODE (Product Code)	CHAR	5	D
PNAME (Product Name)	CHAR	30	D
SCODE (Saleman code)	CHAR	5	D
SALEVAL (Sale value)	NUM	10.2	V
COSTVAL (Cost value)	NUM	10.2	V

Let attribute *CCODE* be a_1, attribute *PCODE* be a_2, attribute *PNAME* be a_3, attribute *SCODE* be a_4, attribute *SALEVAL* be a_5, and attribute *COSTVAL* be a_6. Each candidate dimension (e.g. a_1, a_2, a_3, a_4) is used as base attribute and maps with other attributes of *SALE* data entity. The data sets mapping results follow.

$$\overline{A} = \overline{A_1} \cup \overline{A_2} \cup \overline{A_3} \cup \overline{A_4}$$

$$\overline{A_1} = \begin{Bmatrix} (a_1 \infty a_2),(a_1 \infty a_3),(a_1 \infty a_4),(a_1 \infty a_5),(a_1 \infty a_6),(a_1 \infty a_2 \infty a_3),(a_1 \infty a_2 \infty a_4),(a_1 \infty a_2 \infty a_5), \\ (a_1 \infty a_2 \infty a_6),(a_1 \infty a_3 \infty a_4),(a_1 \infty a_3 \infty a_5),(a_1 \infty a_3 \infty a_6),(a_1 \infty a_4 \infty a_5),(a_1 \infty a_4 \infty a_6),(a_1 \infty a_5 \infty a_6), \\ (a_1 \infty a_2 \infty a_3 \infty a_4),(a_1 \infty a_2 \infty a_3 \infty a_5),(a_1 \infty a_2 \infty a_3 \infty a_6),(a_1 \infty a_3 \infty a_4 \infty a_5),(a_1 \infty a_3 \infty a_4 \infty a_6), \\ (a_1 \infty a_4 \infty a_5 \infty a_6),(a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5),(a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_6),(a_1 \infty a_3 \infty a_4 \infty a_5 \infty a_6), \\ (a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5 \infty a_6) \end{Bmatrix}$$

$$\overline{A_2} = \begin{Bmatrix} (a_2 \infty a_1),(a_2 \infty a_3),(a_2 \infty a_4),(a_2 \infty a_5),(a_2 \infty a_6),(a_2 \infty a_1 \infty a_3),(a_2 \infty a_1 \infty a_4),(a_2 \infty a_1 \infty a_5), \\ (a_2 \infty a_1 \infty a_6),(a_2 \infty a_3 \infty a_4),(a_2 \infty a_3 \infty a_5),(a_2 \infty a_3 \infty a_6),(a_2 \infty a_4 \infty a_5),(a_2 \infty a_4 \infty a_6),(a_2 \infty a_5 \infty a_6), \\ (a_2 \infty a_1 \infty a_3 \infty a_4),(a_2 \infty a_1 \infty a_3 \infty a_5),(a_2 \infty a_1 \infty a_3 \infty a_6),(a_2 \infty a_3 \infty a_4 \infty a_5),(a_2 \infty a_3 \infty a_4 \infty a_6), \\ (a_2 \infty a_4 \infty a_5 \infty a_6),(a_2 \infty a_1 \infty a_3 \infty a_4 \infty a_5),(a_2 \infty a_1 \infty a_3 \infty a_5 \infty a_6),(a_2 \infty a_1 \infty a_4 \infty a_5 \infty a_6), \\ (a_2 \infty a_1 \infty a_3 \infty a_4 \infty a_5 \infty a_6) \end{Bmatrix}$$

$$\overline{A_3} = \begin{Bmatrix} (a_3 \infty a_1),(a_3 \infty a_2),(a_3 \infty a_4),(a_3 \infty a_5),(a_3 \infty a_6),(a_3 \infty a_1 \infty a_2),(a_3 \infty a_1 \infty a_4),(a_3 \infty a_1 \infty a_5), \\ (a_3 \infty a_1 \infty a_6),(a_3 \infty a_2 \infty a_4),(a_3 \infty a_2 \infty a_5),(a_3 \infty a_2 \infty a_6),(a_3 \infty a_4 \infty a_5),(a_3 \infty a_4 \infty a_6),(a_3 \infty a_5 \infty a_6), \\ (a_3 \infty a_1 \infty a_2 \infty a_4),(a_3 \infty a_1 \infty a_2 \infty a_5),(a_3 \infty a_1 \infty a_2 \infty a_6),(a_3 \infty a_2 \infty a_4 \infty a_5),(a_3 \infty a_2 \infty a_4 \infty a_6), \\ (a_3 \infty a_4 \infty a_5 \infty a_6),(a_3 \infty a_1 \infty a_2 \infty a_4 \infty a_5),(a_3 \infty a_1 \infty a_2 \infty a_4 \infty a_6),(a_3 \infty a_1 \infty a_4 \infty a_5 \infty a_6), \\ (a_3 \infty a_1 \infty a_2 \infty a_4 \infty a_5 \infty a_6) \end{Bmatrix}$$

$$\overline{A_4} = \begin{Bmatrix} (a_4 \infty a_1),(a_4 \infty a_2),(a_4 \infty a_3),(a_4 \infty a_5),(a_4 \infty a_6),(a_4 \infty a_1 \infty a_2),(a_4 \infty a_1 \infty a_3),(a_4 \infty a_1 \infty a_5), \\ (a_4 \infty a_1 \infty a_6),(a_4 \infty a_2 \infty a_3),(a_4 \infty a_2 \infty a_5),(a_4 \infty a_2 \infty a_6),(a_4 \infty a_3 \infty a_5),(a_4 \infty a_3 \infty a_6),(a_4 \infty a_5 \infty a_6), \\ (a_4 \infty a_1 \infty a_2 \infty a_3),(a_4 \infty a_1 \infty a_2 \infty a_5),(a_4 \infty a_1 \infty a_2 \infty a_6),(a_4 \infty a_2 \infty a_3 \infty a_5),(a_4 \infty a_2 \infty a_3 \infty a_6), \\ (a_4 \infty a_3 \infty a_5 \infty a_6),(a_4 \infty a_1 \infty a_2 \infty a_3 \infty a_5),(a_4 \infty a_1 \infty a_2 \infty a_3 \infty a_6),(a_4 \infty a_1 \infty a_3 \infty a_5 \infty a_6), \\ (a_4 \infty a_1 \infty a_2 \infty a_3 \infty a_5 \infty a_6) \end{Bmatrix}$$

To identify the relationship of each attribute, the mapping of data sets is executed by ICP (see previous example). The first process involves the elimination of similar data sets from the comparative process, also known as the Classical Set Theory. The next process inspects the mapped data sets and eliminates all with variable attributes. The results follow.

$$\overline{A}_{result} = \begin{Bmatrix} (a_1 \infty a_2),(a_1 \infty a_3),(a_1 \infty a_4),(a_1 \infty a_5),(a_1 \infty a_6),(a_2 \infty a_3),(a_2 \infty a_4),(a_2 \infty a_5),(a_2 \infty a_6),(a_3 \infty a_4), \\ (a_3 \infty a_5),(a_3 \infty a_6),(a_4 \infty a_5),(a_4 \infty a_6),(a_1 \infty a_2 \infty a_3),(a_1 \infty a_2 \infty a_4),(a_1 \infty a_2 \infty a_5), \\ (a_1 \infty a_2 \infty a_6),(a_1 \infty a_3 \infty a_4),(a_1 \infty a_3 \infty a_5),(a_1 \infty a_3 \infty a_6),(a_1 \infty a_4 \infty a_5),(a_1 \infty a_4 \infty a_6),(a_1 \infty a_5 \infty a_6), \\ (a_2 \infty a_3 \infty a_4),(a_2 \infty a_3 \infty a_5),(a_2 \infty a_3 \infty a_6),(a_2 \infty a_4 \infty a_5),(a_2 \infty a_4 \infty a_6),(a_2 \infty a_5 \infty a_6), \\ (a_3 \infty a_4 \infty a_5),(a_3 \infty a_4 \infty a_6),(a_3 \infty a_5 \infty a_6),(a_4 \infty a_5 \infty a_6), \\ (a_1 \infty a_2 \infty a_3 \infty a_4),(a_1 \infty a_2 \infty a_3 \infty a_5),(a_1 \infty a_2 \infty a_3 \infty a_6),(a_1 \infty a_3 \infty a_4 \infty a_5),(a_1 \infty a_3 \infty a_4 \infty a_6), \\ (a_1 \infty a_4 \infty a_5 \infty a_6),(a_2 \infty a_3 \infty a_4 \infty a_5),(a_2 \infty a_3 \infty a_4 \infty a_6),(a_2 \infty a_4 \infty a_5 \infty a_6),(a_3 \infty a_4 \infty a_5 \infty a_6), \\ (a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5),(a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_6),(a_1 \infty a_2 \infty a_4 \infty a_5 \infty a_6),(a_1 \infty a_3 \infty a_4 \infty a_5 \infty a_6), \\ (a_2 \infty a_3 \infty a_4 \infty a_5 \infty a_6),(a_1 \infty a_2 \infty a_3 \infty a_4 \infty a_5 \infty a_6) \end{Bmatrix}$$

Process 3 uses the SQL process to compare the relationship of each attribute. The mapping of data sets executed by Process 3 are composed of $(a_1 \infty a_2)$, $(a_1 \infty a_3)$, $(a_1 \infty a_4)$, $(a_1 \infty a_5)$, $(a_1 \infty a_6)$, $(a_2 \infty a_3)$, $(a_2 \infty a_4)$, $(a_2 \infty a_5)$, $(a_2 \infty a_6)$, $(a_3 \infty a_4)$, $(a_3 \infty a_5)$, $(a_3 \infty a_6)$, $(a_4 \infty a_5)$, and $(a_4 \infty a_6)$. The relation attributes of Process 4 proceeds to classify the actual and relative dimensions. Any indexed key field is immediately tagged as actual dimension, the rest are tagged as relative dimension. Process 5 eliminates the rest of the mapped data sets that are included in the relative dimension attribute. The results follow.

$$\overline{A}_{result} = \begin{cases} (a_1 \infty a_2), (a_1 \infty a_4), (a_1 \infty a_5), (a_1 \infty a_6), (a_2 \leftrightarrow a_3), (a_2 \infty a_4), (a_2 \infty a_5), (a_2 \infty a_6), \\ (a_4 \infty a_5), (a_4 \infty a_6), (a_1 \infty a_2 \infty a_4), (a_1 \infty a_2 \infty a_5), (a_1 \infty a_2 \infty a_6), (a_1 \infty a_4 \infty a_5), \\ (a_1 \infty a_4 \infty a_6), (a_1 \infty a_5 \infty a_6), (a_2 \infty a_4 \infty a_5), (a_2 \infty a_4 \infty a_6), (a_2 \infty a_5 \infty a_6), \\ (a_4 \infty a_5 \infty a_6), (a_1 \infty a_2 \infty a_4 \infty a_5), (a_1 \infty a_2 \infty a_4 \infty a_6), (a_1 \infty a_2 \infty a_5 \infty a_6), \\ (a_1 \infty a_4 \infty a_5 \infty a_6), (a_2 \infty a_4 \infty a_5 \infty a_6), (a_1 \infty a_2 \infty a_4 \infty a_5 \infty a_6) \end{cases}$$

Process 6 tags the mapped data sets as either candidate variables or actual dimensions, and replaces the joint function symbol with the associate function symbol:

$$\overline{A}_{result} = \begin{cases} (a_1 \Rightarrow a_5), (a_1 \Rightarrow a_6), (a_2 \leftrightarrow a_3), (a_2 \Rightarrow a_5), (a_2 \Rightarrow a_6), \\ (a_4 \Rightarrow a_5), (a_4 \Rightarrow a_6), (a_1 \infty a_2 \Rightarrow a_5), (a_1 \infty a_2 \Rightarrow a_6), (a_1 \infty a_4 \Rightarrow a_5), \\ (a_1 \infty a_4 \Rightarrow a_6), (a_1 \Rightarrow a_5, a_6), (a_2 \infty a_4 \Rightarrow a_5), (a_2 \infty a_4 \Rightarrow a_6), (a_2 \Rightarrow a_5, a_6), \\ (a_4 \Rightarrow a_5, a_6), (a_1 \infty a_2 \Rightarrow a_5, a_6), (a_1 \infty a_4 \Rightarrow a_5, a_6), (a_2 \infty a_4 \Rightarrow a_5, a_6), \\ (a_1 \infty a_2 \infty a_4 \Rightarrow a_5, a_6), \end{cases}$$

Process 7 eliminates the mapped data sets which are duplicates:

$$\overline{A}_{result} = \begin{cases} (a_1 \Rightarrow a_5, a_6), (a_2 \leftrightarrow a_3), (a_2 \Rightarrow a_5, a_6), (a_4 \Rightarrow a_5, a_6), \\ (a_1 \infty a_2 \Rightarrow a_5, a_6), (a_1 \infty a_4 \Rightarrow a_5, a_6), (a_2 \infty a_4 \Rightarrow a_5, a_6), (a_1 \infty a_2 \infty a_4 \Rightarrow a_5, a_6) \end{cases}$$

The final process replaces the mapped data sets with data models represented by D as dimension, V as variable, and R as relative dimension:

$D_1.CCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$
$D_2.PCODE \leftrightarrow R_1.PNAME$
$D_2.PCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$
$D_3.SCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$
$D_1.CCODE \infty D_2.PCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$
$D_1.CCODE \infty D_3.SCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$
$D_2.PCODE \infty D_3.SCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$
$D_1.CCODE \infty D_2.PCODE \infty D_3.SCODE \Rightarrow V_1.SALEVAL,\ V_2.COSTVAL$

3.2.3 Clustering Module

While the partitioning module transforms flat data into multidimensional DVR structures, a warehouse database contains a number of relevant data entities and data-sharing capabilities from external sources. Thus, the relationships among relative dimensions of numerous entities must link with associated attributes.

As in partitioning modules, there are two particular steps in the clustering module process: First, the arrangement of grouping relation set is done through the mapping process. Second, the aggregation of the grouping relation sets is done through the External Comparison Process (ECP). The grouping relation set is presented in data set format.

Data sets are arranged in series to verify the relationships in the mapping table, which is composed of base and related data sets. One of these data sets is assigned as a base data set and is compared with other data sets to establish relationships. Once a relationship between base and related data sets is discovered, a new data set is generated (The process is described in the next part of this research). During the process, a particular data set may be removed from the series to avoid repetitive comparison. Another data set is sequentially assigned as the next base data set, and the process continues until there is no base data set left, or until the related data set is empty.

For example, a warehouse database denoted as DB_i has three data entities: *A, B, and C,* assigned into DVR models in a partitioning module. Data entity *A* contains the attribute $\{a_1,\ a_2,\ a_3,\ a_4\}$ and rebuilds into data sets $(a_1 \Rightarrow a_3,\ a_4)$, $(a_2 \Rightarrow a_3,\ a_4)$. Data

Entity B contains the attribute $\{a_1, a_5, a_6\}$ and rebuilds into data sets $(a_1 \Rightarrow a_6)$, $(a_5 \Rightarrow a_6)$. Data entity C contains the attribute $\{a_5, a_7\}$ and rebuilds into data sets $(a_5 \Rightarrow a_7)$. Consequently, the data sets of DB_i are assigned as: $(a_1 \Rightarrow a_3, a_4)$, $(a_2 \Rightarrow a_3, a_4)$, $(a_1 \Rightarrow a_6)$, $(a_5 \Rightarrow a_6)$, $(a_5 \Rightarrow a_7)$. Data set $(a_1 \Rightarrow a_3, a_4)$ is first assigned as base data set and the others as related data sets. The execution of ECP from $(a_1 \Rightarrow a_3, a_4)$ as base data set yields:

$$(a_1 \Rightarrow a_3, a_4) \cup (a_2 \Rightarrow a_3, a_4)$$
$$(a_1 \Rightarrow a_3, a_4) \cup (a_1 \Rightarrow a_6)$$
$$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_6)$$
$$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a\text{-})$$

Similarly, using ECP for $(a_2 \Rightarrow a_3, a_4)$, $(a_1 \Rightarrow a_6)$, $(a_5 \Rightarrow a_6)$, $(a_5 \Rightarrow a_7)$ as base data sets yields the following results (see Table 3.12):

Table 3.12. An Example of a Clustering Module.

Base Data Set	Related Data Set	Mapping Data Set
$(a_1 \Rightarrow a_3, a_4)$	$(a_2 \Rightarrow a_3, a_4)$	$(a_1 \Rightarrow a_3, a_4) \cup (a_2 \Rightarrow a_3, a_4)$
	$(a_1 \Rightarrow a_6)$	$(a_1 \Rightarrow a_3, a_4) \cup (a_1 \Rightarrow a_6)$
	$(a_5 \Rightarrow a_6)$	$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_6)$
	$(a_5 \Rightarrow a\text{-})$	$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a\text{-})$
$(a_2 \Rightarrow a_3, a_4)$	$(a_1 \Rightarrow a_6)$	$(a_2 \Rightarrow a_3, a_4) \cup (a_1 \Rightarrow a_6)$
	$(a_5 \Rightarrow a_6)$	$(a_2 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_6)$
	$(a_5 \Rightarrow a\text{-})$	$(a_2 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a\text{-})$
$(a_1 \Rightarrow a_6)$	$(a_5 \Rightarrow a_6)$	$(a_1 \Rightarrow a_6) \cup (a_5 \Rightarrow a_6)$
	$(a_5 \Rightarrow a\text{-})$	$(a_1 \Rightarrow a_6) \cup (a_5 \Rightarrow a\text{-})$
$(a_5 \Rightarrow a_6)$	$(a_5 \Rightarrow a\text{-})$	$(a_5 \Rightarrow a_6) \cup (a_5 \Rightarrow a\text{-})$
$(a_5 \Rightarrow a\text{-})$	*<End of file>*	*<End of Execution>*

The following processes introduce the supplement ECP algorithms to form new data sets.

PROCESS 1: Referring to the Aggregation Rule, when dimension attributes in data entities are similar, the associate data values of dimension attributes are integrated into one data set.

To simplify the data set, the algorithm aggregates the dispersion of variable attributes where dimension attributes are alike. For example, when data sets ($a_1 \Rightarrow a_3$,a_4) and ($a_1 \Rightarrow a_6$) are represented in the same dimension attribute that is defined in the data dictionary, then variable attributes are integrated into a grouping relation that is set into a new data set ($a_1 \Rightarrow a_3, a_4, a_6$).

PROCESS 2: Any data set that exists in the same data entity skips the comparison process. In Process 7 of the partitioning module, data sets with similar dimension attributes are deleted, retaining only one data set.

Data sets with similar data entities are not compared in this process. Data sets that are segregated by dimension attributes are grouped together to minimize data set duplication. For example, when data sets $(a_1 \Rightarrow a_6)$, $(a_1 \Rightarrow a_3, a_4)$ and $(a_1 \Rightarrow a_3, a_4, a_6)$ are grouped together, the process retains only one data set $(a_1 \Rightarrow a_3, a_4, a_6)$. The rest are deleted.

PROCESS 3: Dimension attributes which are absolutely unrelated to other dimensions ($D_x^i \not\subset D_y^j$) skips the comparison process.

For example, data sets $(a_1 \Rightarrow a_3, a_4, a_6)$ and $(a_2 \Rightarrow a_5)$ are compared. When results prove that data set $(a_1 \Rightarrow a_3, a_4, a_6)$ is not consistent with data set $(a_2 \Rightarrow a_5)$, then the comparison process skips these data sets and proceeds to the next data sets.

PROCESS 4: To drill down the level of granularity, the dimension attribute that is a component of data sets is considered in the grouping relation. To combine related dimensions, the query process $\Gamma_{sum}^{a_x, a_y}(V_{xy})$ is intuitively used to store data values in the data loading process.

Multidimensional databases do not support the logical joining of multiple multidimensional arrays. Its inability to join databases requires the classification of relationships and combinations of any variable before generating the MDDB [10].

For example, dimension attribute a_2 is to be related to data set $(a_2 \Rightarrow a_5)$ in data entity *A*. Furthermore, data set $(a_1 \infty a_2 \Rightarrow a_3)$ is the data entity of *B*. The data sets are thus combined as $(a_2 \Rightarrow a_5) \cup (a_1 \infty a_2 \Rightarrow a_3)$ and the result is a new data set, $(a_1 \infty a_2 \Rightarrow a_3, a_5)$. Data loading will automatically generate an SQL query process that joins data entities *A* and *B*. Using ECP, the resulting sets follow (see Table 3.13).

Table 3.13. Data Sets from ECP.

Base Data Set	*Related Data Set*	*Mapping Data Set*	*ECP Results*
$(a_1 \Rightarrow a_3, a_4)$	$(a_2 \Rightarrow a_3, a_4)$	$(a_1 \Rightarrow a_3, a_4) \cup (a_2 \Rightarrow a_3, a_4)$	-
	$(a_1 \Rightarrow a_6)$	$(a_1 \Rightarrow a_3, a_4) \cup (a_1 \Rightarrow a_6)$	$(a_1 \Rightarrow a_3, a_4, a_6)$
	$(a_5 \Rightarrow a_6)$	$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_6)$	-
	$(a_5 \Rightarrow a\text{-})$	$(a_1 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a\text{-})$	-
$(a_2 \Rightarrow a_3, a_4)$	$(a_1 \Rightarrow a_6)$	$(a_2 \Rightarrow a_3, a_4) \cup (a_1 \Rightarrow a_6)$	-
	$(a_5 \Rightarrow a_6)$	$(a_2 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a_6)$	-
	$(a_5 \Rightarrow a\text{-})$	$(a_2 \Rightarrow a_3, a_4) \cup (a_5 \Rightarrow a\text{-})$	-
$(a_1 \Rightarrow a_6)$	$(a_5 \Rightarrow a_6)$	$(a_1 \Rightarrow a_6) \cup (a_5 \Rightarrow a_6)$	-
	$(a_5 \Rightarrow a\text{-})$	$(a_1 \Rightarrow a_6) \cup (a_5 \Rightarrow a\text{-})$	-
$(a_5 \Rightarrow a_6)$	$(a_5 \Rightarrow a\text{-})$	$(a_5 \Rightarrow a_6) \cup (a_5 \Rightarrow a\text{-})$	$(a_5 \Rightarrow a_6, a\text{-})$
$(a_5 \Rightarrow a\text{-})$	*<End of file>*	*<End of Execution>*	-

Example 3: In examples 2.1 and 2.2, when the data sets are already assigned as DVR models in the partitioning module, the actual dimensions of the *SALEMAN* code and *AREA* code are examined to find associations between them.

The actual variables of *COMMISSION* value and *SALE* value are examined if they can be integrated when actual dimensions are related. The attributes of data entity *A* are transformed into a DVR structure:

-- $D_1'.SCODE \Rightarrow V_1'.SALEVAL, V_2'.COMM$
-- $D_1'.SCODE \leftrightarrow R_1'.SNAME$ (Not mention)
-- $D_2'.ACODE \Rightarrow V_1'.SALEVAL, V_2'.COMM$
-- $D_1'.SCODE \infty D_2'.ACODE \Rightarrow V_1'.SALEVAL, V_2'.COMM$

Additional data sets and their sample data elements from data entities *B*, *C* are assumed to completely form the DVR models:

-- $D_1^2.CCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$
-- $D_2^2.PCODE \leftrightarrow R_1^2.PNAME$(Not mentioned)
-- $D_2^2.PCODE \Rightarrow V_1^2.SALEVAL\ ,\ V_2^2.COSTVAL$
-- $D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$
-- $D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$
-- $D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$
-- $D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$
-- $D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$

The following table presents the sequential arrangement of DVR models and the mapped data sets (see Table 3.14):

Table 3.14. Mapping Data Set of Example 3.

Based and Related Data Set	Mapping Data Set
$\underline{D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM}$	
$D_2^1.ACODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_2^1.ACODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM)$
$D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL.\ V_2^1.COMM$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM)$
$D_1^2.CCODE \Rightarrow V_1^2.SALEVAL.\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL)$
$D_2^2.PCODE \Rightarrow V_1^2.SALEVAL\ ,\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL\ ,\ V_2^2.COSTVAL)$
$D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$	$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL)$

Table 3.14. Mapping Data Set of Example 3 (continued).

Based and Related Data Set	Mapping Data Set
$\underline{D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM}$	
$D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)$
$D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM}$	
$D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	
$D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

Table 3.14. Mapping Data Set of Example 3 (continued).

Based and Related Data Set	Mapping Data Set
$\underline{D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	
$D_3^2.SCODE \Rightarrow V_1^2.SALEVAL. V_2^2.COSTVAL$	$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	
$D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	
$D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	
$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	$(D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$
$\underline{D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$	-

From partitioning module, the twelve data models (Examples 2.1 and 2.2) are rearranged in series called *grouping relation set*. The first data set ($\underline{D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM}$), which is the base data set, is mapped with other data sets until the end of grouping relation set. The next data set, then replaces the previously used based data set,

and so on, until the last data set ($\underline{D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL}$).

The mapped data sets are then processed by ECP to form new data sets (see Table 3.15):

Table 3.15: New Data Set of Example 3.

Mapping data set	New data set from ECP
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)$	-
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)$	-
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_2^2.PCODE \infty D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-

Table 3.15: New Data Set of Example 3 (continued).

Mapping data set	New data set from ECP
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_1^2.CCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	$D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$
$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-

Table 3.15: New Data Set of Example 3 (continued).

Mapping data set	New data set from ECP
$(D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-
$(D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$	-

Initially used as the first base data set, the data set ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$) is compared with the existing grouping relation sets:

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL)$

$(D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM) \cup (D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

Both data sets ($D_1^1.SCODE \leftrightarrow R_1^1.SNAME$) and ($D_2^2.PCODE \leftrightarrow R_1^2.PNAME$) are not examined for relationships because any relative attributes depend on one dimension attribute.

The mapped data set ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) $\cup$ ($D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$) represents the same dimension attribute (*SCODE*). Process 1 aggregates these data sets into new data sets: $D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM,\ V_2^2.COSTVAL$.

Process 2 ignores the mapped data set ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) $\cup$ ($D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) which resulted from same data entity in the partitioning module.

When there is no relationship between dimension attributes $D_1^1.SCODE$ and $D_2^2.PCODE$ in mapped data set ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) $\cup$ ($D_2^2.PCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$), Process 3 skips these data sets.

Finally, the mapped data set ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) $\cup$ ($D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$) is considered. When data set ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) and ($D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$) are compared in Process 4, dimension attribute SCODE is found to be similar to definitions in the data dictionary, the new data set ($D_2^2.PCODE \infty D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM,\ V_2^2.COSTVAL$) is automatically generated.

ECP processes ($D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$) as base data set and yields the following:

$D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM,\ V_2^2.COSTVAL$
$D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM,\ V_2^2.COSTVAL$
$D_2^2.PCODE \infty D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM,\ V_2^2.COSTVAL$

After the clustering module process is completed, the grouping relation set is:

-- $D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$
-- $D_1^1.SCODE \leftrightarrow R_1^1.SNAME$
-- $D_2^1.ACODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$

-- $D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$

-- $D_1^2.CCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_2^2.PCODE \leftrightarrow R_1^2.PNAME$

-- $D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL, V_2^2.COSTVAL$

-- $D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

-- $D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

-- $D_2^2.PCODE \infty D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

-- $D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

-- $D_1^2.CCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

-- $D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

-- $D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

The grouping relation sets of partitioning modules and combined grouping relation sets of clustering modules are represented in the form of logical DVR models, which can be modified at designated interface nodes (Figure 3.2). The resulting DVR structures automatically generate an SQL statement in order to import data values. The process is described in the next section.

3.2.4 Building a Physical *DVR* Model

The three modules (classification, partitioning, and clustering) transform data in DVR structures, which are carefully reviewed for error contamination. In addition to archiving data volumes, analysts may approve missing dimensions or change the relation set of dimensions at any of the designated interface nodes. In the next step, grouping relation sets rearranges the order of dimensions to the satisfaction of the analysts. In order to crosscheck for data errors, random sample data from the relational database verify the achieved *DVR* structures.

(1) Data Model Relocation Process

The order of dimensions affects the variable's layout in a table or report. The first dimension listed in the variable bears the fastest access data, and the last has the slowest.

Assume the combined grouping relation set is $D_1^1.SCODE \infty D_1^2.ACODE \infty D_1^4.PCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$. The dimension attribute *SCODE* appears on the first order following *ACODE* and *PCODE*. Thus, dimension *SCODE* is the first access data to appear in the columns. Dimension *ACODE* next appears in rows, and dimension *PCODE* appears last in the page (Figure 3a).

Assuming that the dimension attribute *PCODE* is desired to be in the column-level (Figure 3b), then the grouping relation set is modified as $D_1^4.PCODE \infty D_1^1.SCODE \infty D_1^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$.

Although the series of dimensions in a grouping relation set are not significant in conceptual view, it is easy to rotate the multidimensional cube.

$D_1^1.SCODE \infty D_1^2.ACODE \infty D_1^3.PCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$

PCODE : A2500

PCODE : A2200

PCODE : A2100

ACODE	SCODE					
	10200		10210		10240	
EAST	200,000	1,000	0	0	0	0
NORTH	0	0	0	0	0	0
SOUTH	0	0	0	0	0	0
WEST	150,000	750	0	0	410,000	2,050

Figure 3a Original grouping relation set

$D_1^4.PCODE \infty D_1^1.SCODE \infty D_1^2.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$

ACODE : EAST

ACODE : NORTH

ACODE : SOUTH

ACODE : WEST

SCODE	PCODE					
	A2100		A2200		A2500	
10200	150,000	750	0	0	0	0
10210	0	0	0	0	650,000	3,250
10240	410,000	2,050	0	0	0	0

Figure 3b Re-order grouping relation set

Figure 3.3. Data Model Relocation Process.

(2) Data Loading and Building the Physical Data Model

The sample data from the warehouse database may be transferred into *DVR* model forms using the import command of *OLAP* tools. These tools organize one portion of data output to test data consistency and accuracy.

To build the physical *DVR* model, *SQL* statement needs to be automatically generated and executed to create data values before it can be automatically downloaded into the *DVR* models.

The appendix section has examples of building a data model and of the data loading process.

3.3 Conclusion

Related literature shows a serious gap in the field of database management, particularly in minimizing human errors in database maintenance. The errors result from the current manual interface process of transforming relationship patterns in a database from a bidimensional to a multidimensional pattern for multidimensional or DVR analysis. This chapter fills the gap by proposing an automated process that identifies, corrects, summarizes, and screens out errors in a large mass of data by using multidimensional relationship programs. This research explains the basic process of creating DVR models validated by propositional functions. Each process is illustrated by examples and each procedure is stated in algorithms.

This research also illustrates how to relocate data models and efficiently load data in MDDB. The procedure designs DVR models in MOLAP via three data mining techniques: classification, partitioning, and clustering modules. Classification modules verify an attribute of data entity as candidate dimension or a variable of multidimensional cubes. Partitioning modules use candidate dimensions as bases for mapping with other attributes of the same data entity to form mapped data sets. This module uses the Internal Comparison Process (ICP) to manipulate data set to validate the DVR model. Finally, the clustering module uses External Comparison Process (ECP) to manipulate DVR models in the partitioning modules in order to group the related dimensions and associated data values.

These modules work together to generate the multidimensional patterns of relationships using logical formulas that classify, map, and group variables in the data sets. The procedure is applicable to warehouse databases for automatic generation of DVR models for MDDB to function.

IV. DVR DATA MODEL DESIGN VIA GRAPH MODEL

4.1 Introduction

The growing complexity of organizational structures result in complex data interrelationships within an information network, rendering inadequate existing data processing routines. Chapter three presents the DVR model automatic generation procedure using three modules: classification, partitioning, and clustering module. This chapter presents the design of creating DVR model via graph representation. The routine automatically transforms relational database in a data warehouse into a multidimensional database.

The routine uses MOLAP applications to generate graphs of the results. The procedures detail how resulting DVR and graph structures are validated by OLAP methodology. After the routine is completed, the physical data in the multidimensional database can still be modified.

Since the user's view of the enterprise's universe is multi-dimensional, the analyst's conceptual view of OLAP models should also be multi-dimensional. This conceptual schema dictates the design and construction of the model structure and of the dimensional calculations, as well as the analysis of the resulting data relationship patterns.

For better visualization, the graph model is used in presenting data. Analysts find it easy to design while users find it easy to understand. One way of viewing data in graph form is the Entity-Relationship (ER) diagram, a tool used in designing data structures. Detailed ER models have attributes and lines that show relationships between sets of data. In this way, it simplifies complex data relationships. The diagram is also used to analyze business requirements as well as the design of the resulting data structure. It uses specialized programming language and requires trained personnel.

The accuracy of conceptual design is vital in building an information system so that it responds fully to user requirements. This research presents a graph model that shows a routine for processing voluminous data from various sources into groups of related dimensions and associated data values. The graph model describes how data entities are collected from a database through three symbols: center nodes, branch nodes, and rectangle nodes. A primary attribute is presented in a center-node. Foreign keys or character format attributes, and associated dimensions are presented in branch-nodes. Numeric format attributes are data values or variables presented in rectangle-node. Curved lines show MDD relationships. Dashes show links between dimensions that support the OLAP drill-down concept. In complex data the lines overlap to rotate and compare information. The finished graph structure fact scheme is reorganized into MDD models of a hypercube in terms of a standard SQL form supported by algorithms.

4.2 Designing Data Model via Graph Model

The definition of graph theory, published by N.L.Biggs, E.K.Lloyd and R.J.Wilson in 1736, is a finite set of dots called vertices (or nodes) connected by links called edges (or arcs). A graph model is simply a set of unordered pairs of distinct elements of vertices called edges. Sometimes a pair of vertices are connected by multiple edge yielding a multi-graph.

The graph model is gratefully applied to various concepts of statistical information, data mining functions, simulation and visualization. The research applies graph model to design DVR model. There are two basic steps going into the creation of a multidimensional (MDD) structure. First, the relational structure is changed into a graph structure. Then, the relational data elements are converted into dimensional fact elements of nodes and rectangles to represent data dimensions. The result is an MDD structure called a hypercube.

In sum, the graph structure is simply a two-dimensional representation of the hypercube. The multidimensionality of the hypercube is illustrated by links that connect the various dimensions and associated data values. The graph structure on research represents this multidimensionality with basic MDD components and relationships between the various groupings. The MDD structure shows dimensions and logical grouping of attributes with a common atomic key relationship.

A variable represents facts or measures. A relative dimension represents datum associated with dimensional attributes. The relationships between the dimension attributes and their associated data values are called grouping relation set.

A mass of data is first converted into a graph model. Then, it is automatically generated into an MDD structure. Steps to the automatic processing of mass data from relational to MDD format follow.

The design of DVR data model from relational data using graph as:

(1) Defining candidate dimensions and variables. This step inspects all data types for indexed keys and verifies the data entity and the candidate dimension of multidimensional cubes. If the attribute is verified as numeric, the data is classified as variables.

(2) Building the DVR data model in graph form.

   (a) Identifying actual and relative dimensions and variables. The candidate dimensions and variables are manipulated using comparative process to be assigned as actual dimensions, variables, and relative dimensions.

   (b) Defining the main dimensions and sub-dimensions. The indexed key field is labeled as main dimensions, the other actual dimensions are labeled as sub-dimensions.

(c) Drawing the graph structure. The relationships of actual dimensions, variables, and relative dimensions are drawn as graphs with symbols, such as circle nodes and boxes, which are connected with lines, dashes, or arrows.

(d) Linking related graph structures. When dimension attributes of various data entities are related, A dotted line is linked to connect the related dimension attributes.

The graph structures are converted into logical DVR data models, the further task is data transformation and data loading into multidimensional database shown in appendix A.

Example 1 presents the fact tables and data of warehouse database including PRODUCT and SALEMAN data entities, this example will be described in the next two sections (example 2 and example 3).

Example 1: In a warehouse database (DB_1) are data entities *PRODUCT* and *SALESMAN* *D* denotes the candidate dimension, *V* denotes the candidate variable, and *R* denotes the relative dimension.

Each attribute a_i of *PRODUCT* a data entity is a candidate dimension denoted as $D_i^{PRODUCT}$. Each attribute a_i of *PRODUCT* data entity is a candidate variable denoted as $V_i^{PRODUCT}$, where i is the attribute of *PRODUCT* data entity. The information in the *PRODUCT* and *SALESMAN* data entities is illustrated in the following tables (see Tables 4.1, 4.2, 4.3).

Table 4.1. Data Entity of PRODUCT Information.

Attribute	Data Type	Indexed Key	Unique
1. Product	Char (8)	Primary	Yes
2. Category	Char (2)	-	-
3. Size	Char (10)	-	-
4. Salesman Code	Char (5)	Foreign	-
5. Manufacturer Code	Char (5)	Foreign	-
6. Customer Code	Char (5)	Foreign	-
7. Shipment Code	Char (2)	Foreign	-
8. Order Date	DMY	-	-
9. Order Qty	Num (6, 0)	-	-
10. Price	Num (10, 2)	-	-

Table 4.2. Data Entity of SALESMAN Information.

Attribute	Data Type	Indexed Key	Unique
1. Salesman Code	Char (5)	Primary	Yes
2. Area	Char(5)	-	-
3. Function	Char(8)	-	-
4. Commission Rate	Num(4,2)	-	-

Table 4.3. Sample Data Elements of Table 4.1.

Product	Cate-gory	Size	Sales-man	Manu-facturer	Custo-mer	Ship-ment	Order Date MM/YY	Order Qty	Price
A001	Audio	Medium	SE001	20M54	C3215	B1	10/1998	4	$185.00
A002	Audio	Large	SE001	20M54	C1201	B2	11/1998	2	$290.00
A002	Audio	Large	SE002	20M54	C3150	B2	11/1998	1	$290.00
T029	TV	Inch29	NR014	21S01	C2145	D1	10/1998	1	$900.00
T021	TV	Inch21	NR014	21S01	C2319	D1	11/1998	3	$550.00
T021	TV	Inch21	SE001	21S01	C1201	B1	11/1998	4	$550.00
T014	TV	Inch14	NR014	21S01	C3215	D1	12/1998	6	$199.00
T014	TV	Inch14	SE002	21S01	C3150	B2	11/1998	5	$199.00
T014	TV	Inch14	WS023	21S01	C1321	C1	10/1998	4	$199.00
V034	VDO	Mono	WS021	21S01	C3124	C1	11/1998	1	$210.00

4.2.1 Defining Candidate Dimension and Variable

The propositional function is an algorithm that describes the selection process which automatically identifies dimensions and variables in a great mass of data.

A set of candidate dimensions must be verified before they are classified as actual dimension. The process entails careful comparison of the algorithms' various attributes.

A variable is a fact, or measure, usually stored as a numeric field [172]. Several variables are contained within each group of data, and it is important to identify the most relevant ones, and to screen out the irrelevant variables in order to prevent errors in data analysis. This is also done automatically.

The term dimension refers to the logical grouping of attributes with a common atomic key relationship [172]. This is a subject-oriented grouping that includes product, location and time.

(1) Identifying Candidate Dimensions

Each data group has a large number of dimensions. Among these, and the useful data are called actual dimensions, identified using two automatic programs that first finds candidate dimensions, then screens and labels them as actual dimensions.

(a) Classification. When an attribute in a data entity is verified as part of an MDD model, the attribute is automatically classified as candidate dimension.

The propositional function [97] specifies that any attribute x of a data format is classified as a candidate dimension if the attribute is an index key field or its data is expressed as a character or a date. Thus,

$(\forall x)\left(I_x \cup \left(\delta_{cx} \cup \delta_{tx}\right)\right) \Rightarrow D_x^i$ where

I_x is the index key format, normally stored as a key field,

δ_x is the type of data property which is equivalent to a character format (denoted as c) or date/time format (denoted as t), and D_x^i is the attribute x within the dimension of data entity i, when $i > 0$.

(b) Reclassification. Second, any candidate dimension is automatically reclassified as actual dimension using a comparative algorithm. This

procedure is described in Section 4.4, which outlines the steps in building the DVR data model.

(2) Identifying Candidate Variables

After the candidate dimensions are identified, the remaining attributes are screened for candidate variables. The propositional function of identifying candidate variables follows.

Any attribute x is defined as a candidate variable if its data type is numeric, whether in binary, integer, or decimal form:

$$(\forall x)(\delta_{nx}) \Rightarrow V_x^i \text{ where}$$

δ_n is data type in numeric format (denoted by "n"), and

V_x^i is attribute x within the variable of data entity i, where $i > 0$.

The term relative dimension refers to the description of the associated data of a dimension. It defines the relationships of an ordinary attribute to a grouping relation. The first step of defining the relative dimension is to convert it into a candidate dimension. The next step is to repeatedly predicate the characteristic of actual dimension and relative dimension (Section 3.2).

Example 2: Defining candidate dimensions and variables of the data entity *PRODUCT*.

Table 4.1 shows that the attribute *PRODUCT* is the primary key field of the data entity. This is thus identified as candidate dimension.

Other attributes (*SALESMAN, MANUFACTURER, CUSTOMER, and SHIPMENT*) are the foreign key fields of the data entity. These attributes are therefore identified as candidate dimensions corresponding to the *PRODUCT* attribute.

The attributes of *CATEGORY, SIZE,* and *ORDER DATE* are labeled as candidate dimensions because they are in character or date formats.

The attributes of *ORDER QUANTITY* and *PRICE* are classified as candidate variables because they are in numeric format. The result is:

$$D^{PRODUCT}_{product}, D^{PRODUCT}_{salesman}, D^{PRODUCT}_{manufacturer}, D^{PRODUCT}_{customer}, D^{PRODUCT}_{shipment},$$
$$D^{PRODUCT}_{category}, D^{PRODUCT}_{size}, V^{PRODUCT}_{orderquantity}, V^{PRODUCT}_{price}$$

4.2.2 Building the Graph Model of the MDD Structure

The previous section discussed the automated process of identifying candidate dimensions and variables from a mass of high-volume data. This section presents the process of changing candidate dimensions and variables into actual dimensions and variables.

The MDD is a finite set of grouping relations. A grouping relation is a combination of the attributes of dimension, variable, and/or relative dimension. It contains dimension such as base attributes that carry useful information. The variable attributes and relative dimension are derived from the base attributes and are related to associated attributes.

The candidate dimension, which is the primary index of a data entity, is labeled as main dimension, the starting point of the Graph Model. The starting point of a tree structure is also the main dimension, which is represented by a center node. From this, other candidate dimension attributes (sub-dimensions) follow in the form of smaller circles. For instance, elements of Table 4.1 are shown in Figure 4.1 as lines that delineate relationships between the main dimension *PRODUCTS* and the sub-dimension's attributes. There are no relationships between the sub-attributes.

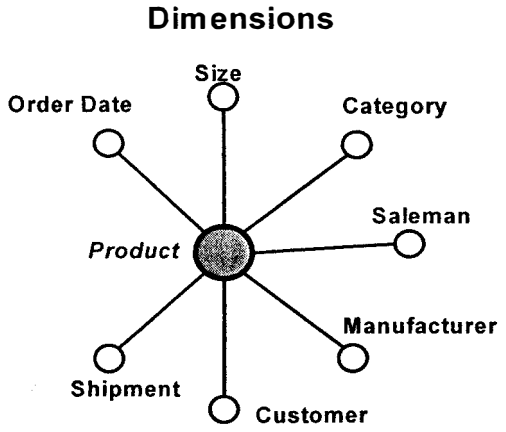


Figure 4.1. Sample Graph Model Form.

This example shows the advantages of multidimensional over bidimensional data analysis. The columnar form of Table 4.1 and the Graph Model in Figure 4.1 contain the same data. However, the multidimensional graph model of Figure 4.1 shows the many relationships between elements, which are vaguely presented on the bidimensional Table 4.1.

The greater the number of relationships between data attributes, the more comprehensive the information will be in an MDD structure, and this makes decision-making a more precise process.

(1) Identifying Actual DVRs

After automatically categorizing data entities, thorough comparison must be made. This repeated predication is important for further verification of the candidate dimensions and variables, as well as DVRS (actual dimensions, variables, and relative dimensions).

The comparative process follows:

(a) Select one data entity from the warehouse database.

(b) From the data entity, select and label one candidate dimension as primary index key.

(c) Use this primary index key to start a data map.

(d) Map the primary index key with the other attributes in the data entity.

(e) Use a comparative algorithm to manipulate the characteristic of each attribute.

(f) Program the algorithm to identify each attribute in each data set as either actual dimension, variable, or relative dimension. Commercial software such as Personal Express Tool™ or Metacube™ may be used instead.

(g) Execute the program.

Data is now ready to be analyzed for relationships among the attributes of one data entity. The Graph Model incorporates the element of relationship between the primary and other attributes: the small white circles indicate sub-dimensions while the black circles indicate relative dimensions (Section 5.2).

Comparative Algorithm:

At each data mapping:

- Initially, set VL_1, VL_2 to empty.
- Count data values of *first attribute* (a_1) of data mapping into VL_1.
- Do while data mapping is not empty
  - If *second attribute* (a_2) of data mapping is candidate dimension.
    - -- Performed *SQL* process of second attribute
      - --- Count data values into VL_2
      - If VL_1 equal to VL_2
      - ---- Set a_2 to be relative dimension.
      - Else ---- Set a_2 to be actual dimension.
    - Else -- Performed *SQL* process of second attribute
      - If VL_1 equal to VL_2
      - ---- Set a_2 to be relative dimension.
      - Else --- Set a_2 to be actual variable
  - VL_2 is set to empty.
  - End Do

End Algorithm

Example 3: A comparative algorithm program for identifying Actual DVRs.

(1) Extract the file *PRODUCT* data entity from the warehouse database.

(2) From the file's many attributes, select $D_{product}$ as the primary index key.

(3) Compare Dproduct with the other attributes.

(4) Map the data as follows: $\{D_{product} \cup D_{salesman}\}$, $\{D_{product} \cup D_{manufacturer}\}$, $\{D_{product} \cup D_{customer}\}$, $\{D_{product} \cup D_{shipment}\}$, $\{D_{product} \cup D_{category}\}$, $\{D_{product} \cup D_{size}\}$, $\{D_{product} \cup V_{orderquantity}\}$, $\{D_{product} \cup V_{price}\}$

(5) Execute the comparative algorithm.

This program executes a routine that counts the item number of $D_{product}$ attribute by aggregate function (e.g. *sum, avg, min, count*) into variable VL_1. The routine is executed until the data mapping set is empty. At the first data mapping set, the comparative algorithm represents the difference between $D_{product}$ and $D_{salesman}$ as candidate dimension. The $D_{salesman}$ attribute is counted by aggregate function of the *SQL* process and recorded into variable VL_2.

(1) Identify actual and relative elements.

The program executes a routine that compares the variables VL_1 and VL_2. If the results show absolute dissimilarity, then the $D_{salesman}$ attribute is labeled as actual dimension. If the results show similarity, then, the attribute is labeled as relative dimension. The program executes the routine to process the candidate dimensions of *MANUFACTURER, CUSTOMER*, and *SHIPMENT* with the *PRODUCT* attribute and then label them as actual dimensions.

If the Step 6 routine shows a similarity of relationships, the candidate dimensions are labeled as *relative dimension*. If dissimilar, they are labeled as *actual dimension*. When the candidate variables of *PRICE* are found to be similar to *PRODUCT, PRICE* is then labeled as a relative dimension. The candidate variable of *ORDER QUANTITY* is automatically labeled as actual variable.

(2) Defining Main and Sub-dimensions

Figure 4.1 shows the relationship between main and sub dimensions, both labeled as actual dimension. An actual dimension or a primary index key is labeled as main dimension. The other actual dimensions are labeled as sub-dimensions.

For instance, a $D_{product}$ attribute is a primary index key of the data entity labeled as *PRODUCT*. The process labels *PRODUCT* attribute as main dimension, the finest aggregation of the tree structure and represented by a large circle. On the other hand, *SALESMAN, MANUFACTURER, CUSTOMER, SHIPMENT*, and *ORDER DATE* attributes are labeled as sub-

dimension represented by tiny circles that are attached to the major circle. At this step, the routine may be programmed as follows:

DEFINE product DIMENSION TEXT WIDTH 8
DEFINE salesman DIMENSION TEXT WIDTH 5
DEFINE manufacturer DIMENSION TEXT WIDTH 5
DEFINE customer DIMENSION TEXT WIDTH 5
DEFINE shipment DIMENSION TEXT WIDTH 2
DEFINE order_date DIMENSION TEXT WIDTH 10

This is an example of creating dimension attributes in multidimensional database via Personal Express Tools™. Small, black circles connected solely to the major circle represent relative dimension attributes. *SIZE, PRICE*, and *CATEGORY* attributes are linked directly to the primary key, *PRODUCT*. The program may be compiled as follows:

DEFINE size VARIABLE TEXT <product>
DEFINE price VARIABLE DECIMAL <product>
DEFINE category VARIABLE TEXT <product>

(3) Drawing the Graph Model

MDD models are drawn as graphs with symbols, such as circle nodes and boxes, which are connected with lines, dashes, or arrows. A large circle shows the main dimension as a center node in the Graph Model. Sub-dimensions are shown as branch nodes represented as small circles. Relative dimensions are solid black and connected solely to the center node; actual dimensions are white and may be connected to either the center node or to branch-nodes.

Actual variables are boxes labeled with attribute names in numeric format and linked solely with the center node. Measurements, in boxes linked to the actual variables, are separated from the actual variable box

since they are formulas, and not physical data values, which may be stored in actual variable boxes. An example follows in Figure 4.2.

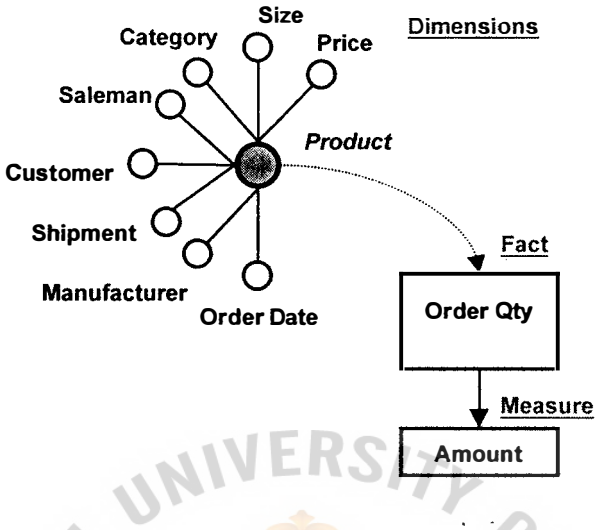


Figure 4.2. Graph Model of PRODUCT Data Entity.

As a result of the multiplication of the attributes *ORDER QUANTITY* and *PRICE,* assume that the measurement of the attribute *AMOUNT* is included in this graph. The graph shows a box with the attribute name linked by a related actual variable. This is an example of a graph model utilizing only bidimensional (2D) relationships. Here, the *SALESMAN* branch node and the *PRODUCT* center node represent only a row-and-column relationship associated with a tabulated form of quantity and price value.

However, a graph model can be made to show an MDD view simply by connecting lines into the desired nodes. The following model shows a 2D graph adapted into a 3D graph.

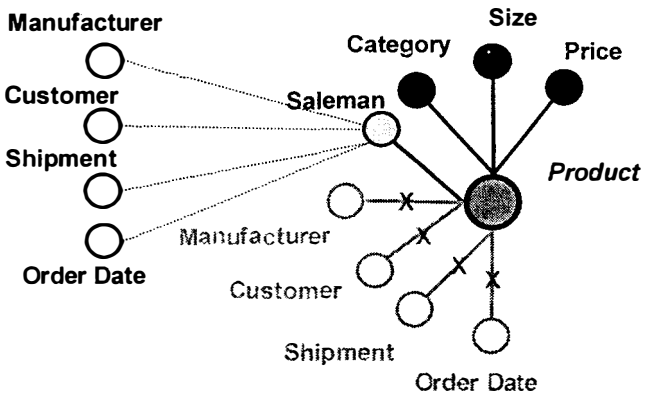


Fig. 4.3a. Analyzing data in 3D in the SALESMAN dimension.

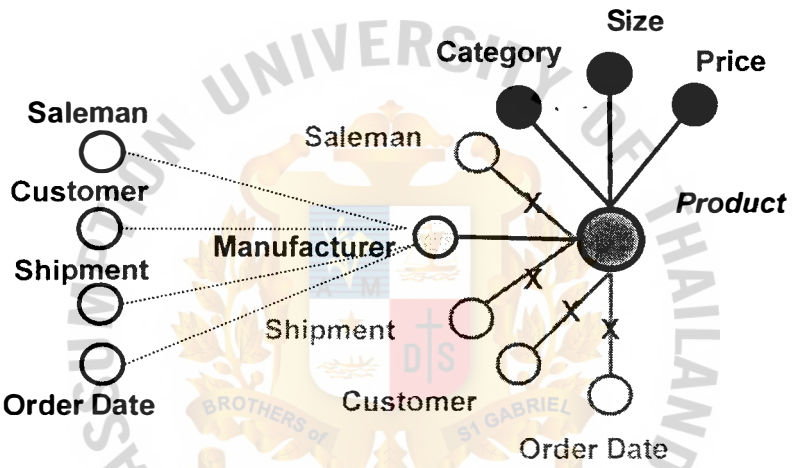


Fig. 4.3b. Analyzing data in 3D in the MANUFACTURER dimension.

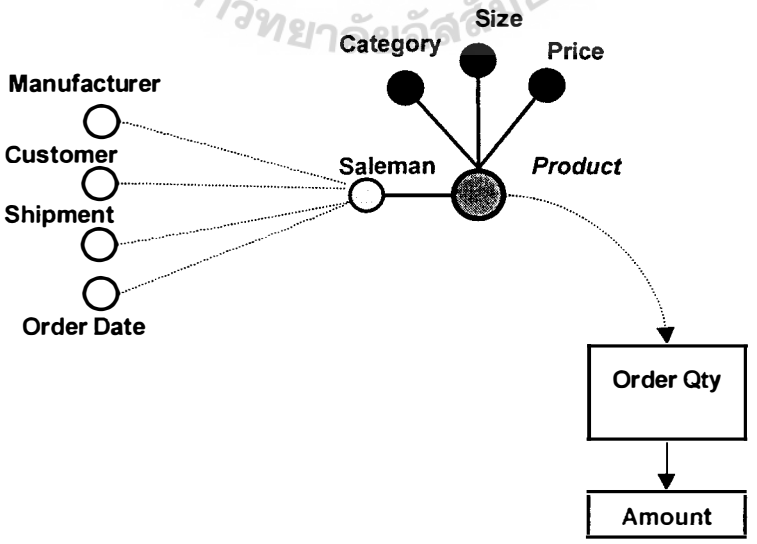


Fig. 4.3c. A 3-D Graph Model.

Figure 4.2 uses a 2D graph model, similar to a table. The result of ORDER QTY is summarized by $\{D_{product} \cup D_{salesman}\}$, $\{D_{product} \cup D_{manufacturer}\}$, $\{D_{product} \cup D_{customer}\}$, $\{D_{product} \cup D_{shipment}\}$, and $\{D_{product} \cup D_{orderdate}\}$ (see Figure 4.3a).

Figures 4.3a and 4.3b show similar results by simply shifting the focus of analysis from *SALESMAN* to *MANUFACTURER*. The multidimensionality of the data analysis is in the flexibility of shifting focus of analysis as desired.

Figure 4.3c shows the summary of *ORDER QUANTITY* as processed by $\{D_{product} \cup D_{salesman} \cup D_{manufacturer}\}$, $\{D_{product} \cup D_{salesman} \cup D_{customer}\}$, $\{D_{product} \cup D_{salesman} \cup D_{shipment}\}$, and $\{D_{product} \cup D_{salesman} \cup D_{orderdate}\}$. The results of processing AMOUNT and the ORDER QUANTITY are the same.

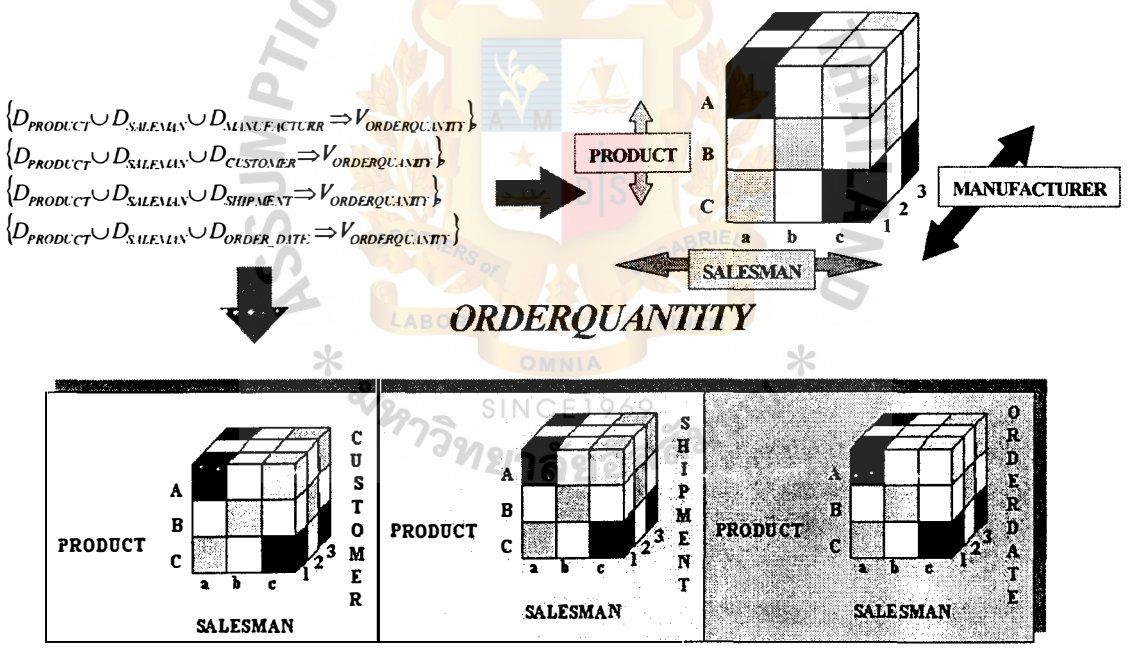


Figure 4.4. A Hypercube Data Structure.

Figure 4.4 shows Figure 4.3c in hypercube. The data structure of variable *ORDERQUANTITY* is generated into four hypercubes via this algorithm:

$\{D_{product} \cup D_{salesman} \cup D_{manufacturer}\}$,
$\{D_{product} \cup D_{salesman} \cup D_{customer}\}$,
$\{D_{product} \cup D_{salesman} \cup D_{shipment}\}$, and
$\{D_{product} \cup D_{salesman} \cup D_{orderdate}\}$.

Variable Process: To transform a graph model into an MDD model using Personal Express Tool™, the variable routine is:

DEFINE orderqty_01 *VARIABLE* <product salesman manufacturer>
LD Orderqty of product-salesman-manufacturer dimension
DEFINE orderqty_02 *VARIABLE* <product salesman customer>
LD Orderqty of product-salesman-customer dimension
DEFINE orderqty_03 *VARIABLE* <product salesman shipment>
LD Orderqty of product-salesman-shipment dimension
DEFINE orderqty_04 *VARIABLE* <product salesman orderdate>
LD Orderqty of product-salesman-orderdate dimension

Measurement Process: To transform a graph model into an MDD model using Personal Express Tool™, the measurement routine is:

DEFINE amount_01 *FORMULA* orderqty_01*Price
LD amount of product-salesman-manufacturer dimension
DEFINE amount_02 *FORMULA* orderqty_02*Price
LD amount of product-salesman-customer dimension
DEFINE amount_03 *FORMULA* orderqty_03*Price
LD amount of product-salesman-shipment dimension
DEFINE amount_04 *FORMULA* orderqty_04*Price
LD amount of product-salesman-orderdate dimension

Creating an MDD structure based on the graph model as shown in Figure 4.3, the researcher finds that a 3D graph model is insufficient to completely analyze the data. In such a case, the model may be reorganized into as many dimensions as desired.

4.2.3 Linking Related Graph Models

Related dimension attributes of various data may use dotted lines to connect related dimension attributes. An example follows.

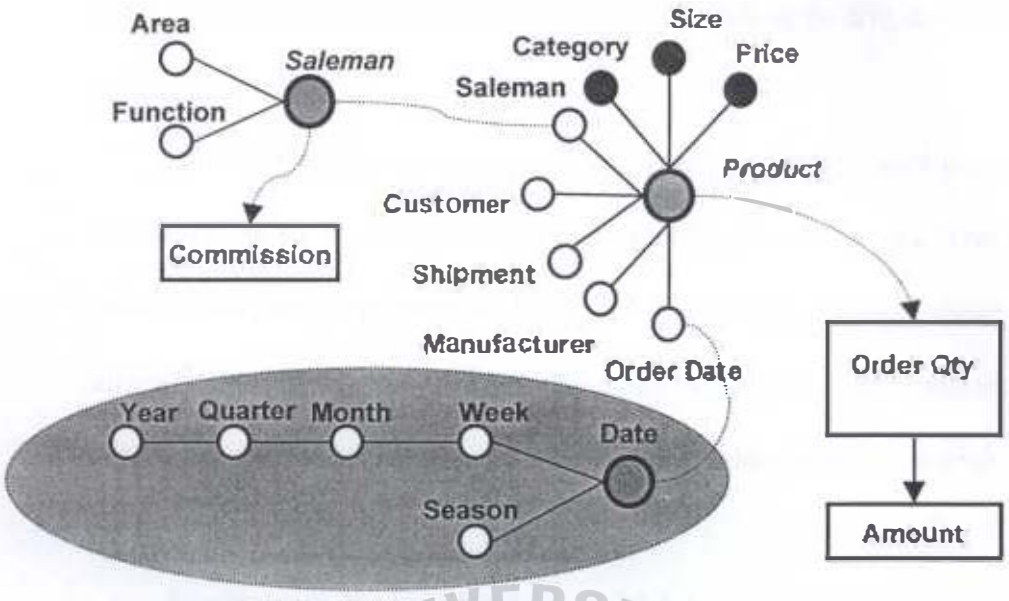


Figure 4.5. Linking Related Graph Structures.

There are three graph structures in Figure 4.5: *PRODUCT*, *SALESMAN*, and *DATE*. In Table 4.2 the graph with the base attribute *SALESMAN* is associated with the same attribute in the *PRODUCT* graph. The connection of the two graph models result in broader dimensional analysis.

For example, management may want to know how much commission came out of the purchases of customer *X*. The analysis is completed when two graph models are combined: $\{D_{product} \cup D_{salesman} \cup D_{customer} \cup D_{orderdate} \Rightarrow V_{commission}\}$.

(1) Additional Aids

Other multidimensional views (*e.g.* roll-ups, drill-downs) are useful in presenting levels of hierarchical data, as they create attributes for each level on the hierarchy. For instance, an *ORDER DATE* attribute may be defined into levels of *WEEK*, *MONTH*, *QUARTER*, and *YEAR*. The attribute *AREA* may be divided into geographical levels as *CONTINENT*, *COUNTRY*,

REGION, CITY. Users are able to concurrently roll-up or drill down through multiple dimensions.

The MDD data structure can incorporate these additional procedures and existing databases can be processed using these add-ons. The MDD model can be reprogrammed to automatically process these new dimensions. This not only widens the range of analysis, it also allows local programmers to utilize routines developed elsewhere to enrich their processing functions.

For example, the year 1998 of *ORDER DATE* attribute is connected to year 1998 in *DATE* data entity. The *QUARTER* sublevel of year 1998 is divided *into Q198, Q298, Q398,* and *Q498*. The *MONTH* sublevel of *Q198* is divided into *JAN98, FEB98,* and *MAR98*, all embedded as aggregation of detailed levels and summed up to the highest level as follows (see Figure 4.6):

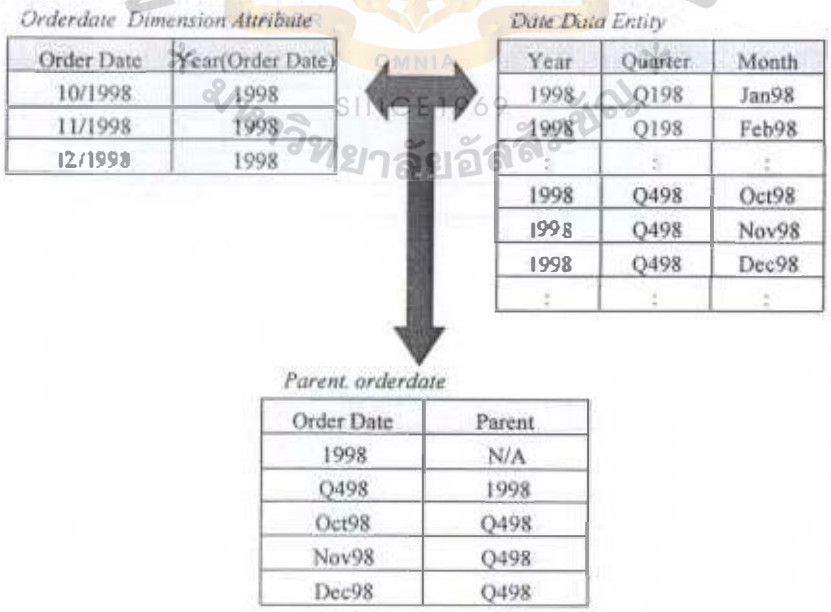


Figure 4.6. Combining Order Date and Date Data Entity.

The routine may be programmed thus:

DEFINE parent.orderdate RELATION orderdate <orderdate>
CALL execution_1
ROLLUP ordersty_04 OVER orderdate USING parent.orderdate
ROLLUP amount_04 OVER orderdate USING parent.orderdate

The system will automatically aggregate values of actual data to the higher level, as follows (see Table 4.4):

Table 4.4. The Example of Order Date.

Shipment	1998	Q498	Oct98	Nov98	Dec98
B1	$2940.00	$2940.00	$740.00	$2200.00	0.00
B2	$1865.00	$1865.00	0.00	$1865.00	0.00
C1	$1006.00	$1006.00	$796.00	$210.00	0.00
D1	$3744.00	$3744.00	$900.00	$1650.00	$1194.00

Example 4: In Example 1 of chapter 3, there are two data entities: *SALES* and *COST* (see Table 4.5, 4.6). To identify the characteristics of data structure, the candidate dimension and variable phases are first identified.

Table 4.5. SALE Data Entity.

Attribute Name	Data Type	Length
SCODE (Saleman code) [Index key]	CHAR	5
SNAME (Saleman Name)	CHAR	30
ACODE (Area code) [Foreign key]	CHAR	3
SALEVAL (Sale value)	NUM	10.2
COMM (Commission)	NUM	10.2

Table 4.6. COST Data Entity.

Attribute Name	Data Type	Length
CCODE (Customer code) [Index key]	CHAR	6
PCODE (Product Code) [Index key]	CHAR	5
PNAME (Product Name)	CHAR	30
SCODE (Saleman code) [Foreign key]	CHAR	5
SALEVAL (Sale value)	NUM	10.2
COSTVAL (Cost value)	NUM	10.2

In the *SALE* data entity, *SCODE*, *SNAME*, and *ACODE* attributes are labeled as candidate dimensions while *SALEVAL* and *COMM* attributes are labeled as candidate variables. In the *COST* data entity, *CCODE*, *PCODE*, *PNAME*, and *SCODE* attributes

are labeled as candidate dimensions while *SALEVAL* and *COSTVAL* attributes are labeled as candidate variables. *SCODE* is structured as an index key field of *SALE* data entity and labeled as a candidate dimension. In the *SNAME* attribute, the data type is in character format, so it is labeled as a candidate dimension. *ACODE*, *CCODE*, *PCODE*, and *PNAME* attributes of both data entities are processed the same way as *SCODE* and *SNAME*. All are labeled as candidate dimensions. Meanwhile, *SALEVAL*, *COSTVAL*, and *COMM* attributes become candidate variables because of the numeric data. The results are generated as tables below, where *D* means candidate dimension and *V* means candidate variable.

Table 4.7. SALE Data Entity and DVR Indicator.

Attribute Name	Data Type	Length	DVR
SCODE (Saleman code) [Index key]	CHAR	5	D
SNAME (Saleman Name)	CHAR	30	D
ACODE (Area code) [Foreign key]	CHAR	3	D
SALEVAL (Sale value)	NUM	10.2	V
COMM (Commission)	NUM	10.2	V

Table 4.8. COST Data Entity and DVR Indicator.

Attribute Name	Data Type	Length	DVR
CCODE (Customer code) [Index key]	CHAR	6	D
PCODE (Product Code) [Index key]	CHAR	5	D
PNAME (Product Name)	CHAR	30	D
SCODE (Saleman code) [Foreign key]	CHAR	5	D
SALEVAL (Sale value)	NUM	10.2	V
COSTVAL (Cost value)	NUM	10.2	V

The above tables (Table 4.7, 4.8) show the resulting DVR indicators of each attribute, all translated in data model structures using the following algorithm:

$$D^{SALE}_{SCODE}, D^{SALE}_{SNAME}, D^{SALE}_{ACODE}, V^{SALE}_{SALEVAL}, V^{SALE}_{COMM}$$
$$D^{COST}_{CCODE}, D^{COST}_{PCODE}, D^{COST}_{PNAME}, D^{COST}_{SCODE}, V^{COST}_{SALEVAL}, V^{COST}_{COSTVAL}$$

Identifying DVR: The steps to identifying the actual dimension, variable, and relative dimension using the comparative process follow.

Choose one data entity from the warehouse database. Process *SALE* data entity, followed by *COST* data entity.

From the data entity, choose one candidate dimension as primary index key. At *SALE* data entity, *SCODE* attribute is selected because its data structure is a primary indexed key field of *SALE* data entity. At *COST* data entity, both *CCODE* and *PCODE* attributes are selected for the same reason.

Use this primary index key field to start a data map. Map the data in two data entities: *SALE* data entity and *COST* data entity.

Map the primary index key with the other attributes in the data entity. *SCODE* attribute is assigned as base and mapped with other attributes within *SALE* data entity. The following algorithm processes items in *SALE* data entity:

$$DM_{SALE} = \begin{Bmatrix} D_{SCODE} \cup D_{SNAME}, D_{SCODE} \cup D_{ACODE}, \\ D_{SCODE} \cup V_{SALEVAL}, D_{SCODE} \cup V_{COMM} \end{Bmatrix}$$

The following algorithm processes items in *COST* data entity:

$$DM_{COST} = \begin{Bmatrix} D_{CCODE} \cup D_{PCODE}, D_{CCODE} \cup D_{PNAME}, D_{CCODE} \cup D_{SCODE}, \\ D_{CCODE} \cup V_{SALEVAL}, D_{CCODE} \cup V_{COSTVAL}, D_{PCODE} \cup D_{PNAME}, \\ D_{PCODE} \cup D_{SCODE}, D_{PCODE} \cup V_{SALEVAL}, D_{PCODE} \cup V_{COSTVAL} \end{Bmatrix}$$

Where DM_x is Data Mapping of x data entity.

(1) Use a comparative algorithm to manipulate the characteristics of attributes.

This step reveals that the number of unique items in dimension *SCODE* are equal to the number of unique items in dimension *SNAME* within the *SALE* data entity. This proves that dimension *SNAME* is related to dimension *SCODE*. Therefore, dimension *SNAME* is labeled as a relative dimension *SNAME*.

Mapping *PCODE* and *PNAME* shows a number of similarities, so *PCODE* is assigned as relative dimension to *PNAME*. This is a process that identifies the real DVRs in a database.

(2) The following program is executed to create both graphic and DVR data structures. The actual and relative dimensions are first defined into a multidimensional database, where relationships are modified if necessary, and then variables are generated.

DEFINE scode DIMENSION TEXT WIDTH 5
LD Salesman code
DEFINE acode DIMENSION TEXT WIDTH 3
LD Area code
DEFINE ccode DIMENSION TEXT WIDTH 6
LD Customer code
DEFINE pcode DIMENSION TEXT WIDTH 5
LD Product code
DEFINE sname VARIABLE TEXT <scode>
LD Salesman Name
DEFINE pname VARIABLE TEXT <pcode>
LD Product Name

The following graph presentation shown in Figure 4.7 introduces a DVR data model of *SALE* and *COST* data entities.

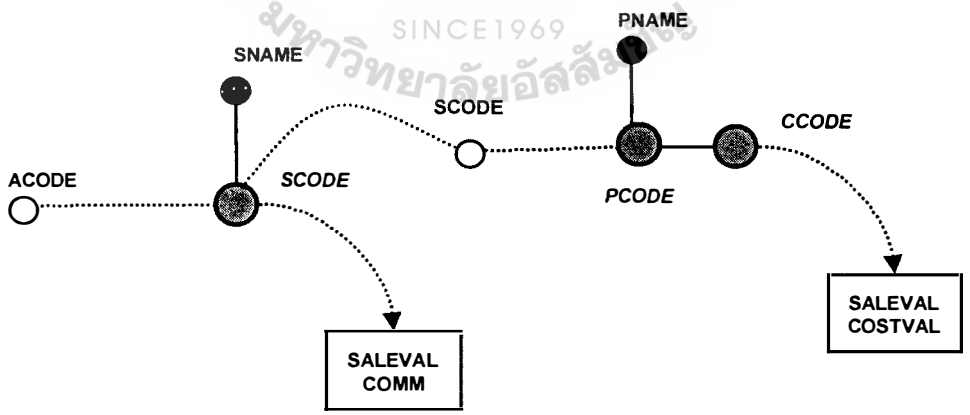


Figure 4.7. SALE and COST Data Entities in Graph Form.

This routine checks the descriptions in metadata (or data directory) and links *SCODE* in *SALE* and *SCODE* in *COST*. This process searches data connections in one data entity to others. The data structure is generated as a graph, using this algorithm:

$\{D_{SCODE} \cup D_{ACODE} \Rightarrow V_{SALEVAL}, V_{COMM}\}$
$\{D_{SCODE} \leftrightarrow R_{SNAME}\}$
$\{D_{CCODE} \cup D_{PCODE} \cup D_{SCODE} \Rightarrow V_{SALEVAL}, V_{COSTVAL}\}$
$\{D_{PCODE} \leftrightarrow R_{PNAME}\}$
$\{ D_{CCODE} \cup D_{PCODE} \cup D_{SCODE} \cup D_{ACODE} \Rightarrow V_{SALEVAL}, V_{COSTVAL}, V_{COMM} \}$

The following items are Oracle Personal Express commands to create variables of a data structure $\{ D_{CCODE} \cup D_{PCODE} \cup D_{SCODE} \cup D_{ACODE} \Rightarrow V_{SALEVAL}, V_{COSTVAL}, V_{COMM} \}$:

DEFINE SALEVAL VARIABLE <CCODE PCODE SCODE ACODE>
LD Sale Values Information

DEFINE COMM VARIABLE <CCODE PCODE SCODE ACODE>
LD Commission Values Information

DEFINE COSTVAL VARIABLE <CCODE PCODE SCODE ACODE>
LD Cost Values Information

An example of an algorithm for loading relational format data from a warehouse into a multidimensional database is as follows:

SQL Process: Select *A.CCODE, A.PCODE, A.SCODE, B.ACODE*, sum (*A.SALEVAL*), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.CCODE, A.PCODE, A,SCODE, B.ACODE*
Order by *A.CCODE, A.PCODE, A,SCODE, B.ACODE*

4.3 Conclusion

This chapter presents a graph model that shows a routine for processing voluminous data from various sources into graph of related dimensions and associated data values. The routine is also expressed by fact schemes that integrate relative information and external sources, as well as extends the algorithm to build DVR patterns in standard SQL form. This schema is based on the basic Multidimensional Data Model elements. This model uses three symbols: a center node represents the fastest variable dimension, the branch nodes represent associate dimension attributes, and rectangles represent data value indexing and measurements.

V. APPLICATION TO ENGINE COST SYSTEM

5.1 Introduction

A critical aspect of the airline business is its technical department, where aircraft maintenance, a major concern, is conducted in four sections: light aircraft, heavy aircraft, engine, and component maintenance. To manage engine cost, top-management planning is required for areas such as engine module per aircraft, man-hours, usage of engine cycle per flight periods, and long-term investment plans. All are done using current Management Information System (MIS). However, the increasing complexity of organizational structure result in more complex data interrelationships that renders inadequate existing data processing routines. This research presents a program that not only creates more dimensional data storage structures, but also wider and more fluid capabilities for data analysis.

Admittedly, an airline company needs to upgrade its data structure and analytical procedures, but it cannot afford to suspend operations or support changes to the detriment of its business operations. The typical project duration is one year, which includes preliminary study, data collection, data analysis, design, programming and implementation. Due to document analyses and data descriptions, the data collection phase takes the longest time. To conserve time and to reduce duplication, design and programming work are distributed into sub-workgroups.

The bulk of the activity is centered on reprogramming the data access layer and begins with the organization of data structures using DVR data models. In this layer, the DVR data model is automatically generated by the procedure, which uses algorithms to map and aggregate data (Chapter 3). The DVR data models are then converted into physical data structures. Finally, the warehouse data volumes are loaded into the physical data structure of the multidimensional database.

Chapter Five concludes the discussion on automatic generation of multidimensional database structures by applying the program to analyze the database of a technical department in an airline company. More specifically, engine cost is used as a pilot subject to test the DVR data model as well as the design process. This chapter explains the technical department's data sources, how data are delivered into its operational data warehouse, how the database is cleansed, the nomenclature of attributes, and other tasks essential to the procedure.

5.2 Data Sources

An airline's engine maintenance department's database includes engine cost, engine cycles per flight hours, manpower, as well as special services for engine maintenance. This research focuses on the database relevant to engine costing in particular. The fact attributes of engine cost comes from various departments within the airline company (e.g. Materials, Production, Accounting) as well as from external sources (e.g. IATA, SITA, Boeing, and Airbus databases). The automatic data feeding process uses the IBM Mainframe platform.

However, not all the incoming data are useful to the engine maintenance department. To screen data, a routine is programmed to allow an airline executive (i.e., the vice-president for technical affairs) to specify required attributes of a criterion. After the data is screened, the data entities are integrated using the joint function (∞). A software program (tapestry product) then clears the data for inconsistencies. The cleared data is stored into the current data level of the department's database.

The E/R scheme shown in Figure 5.1 shows how engine cost data, particularly engine structures and maintenance, is gathered, screened, and stored.

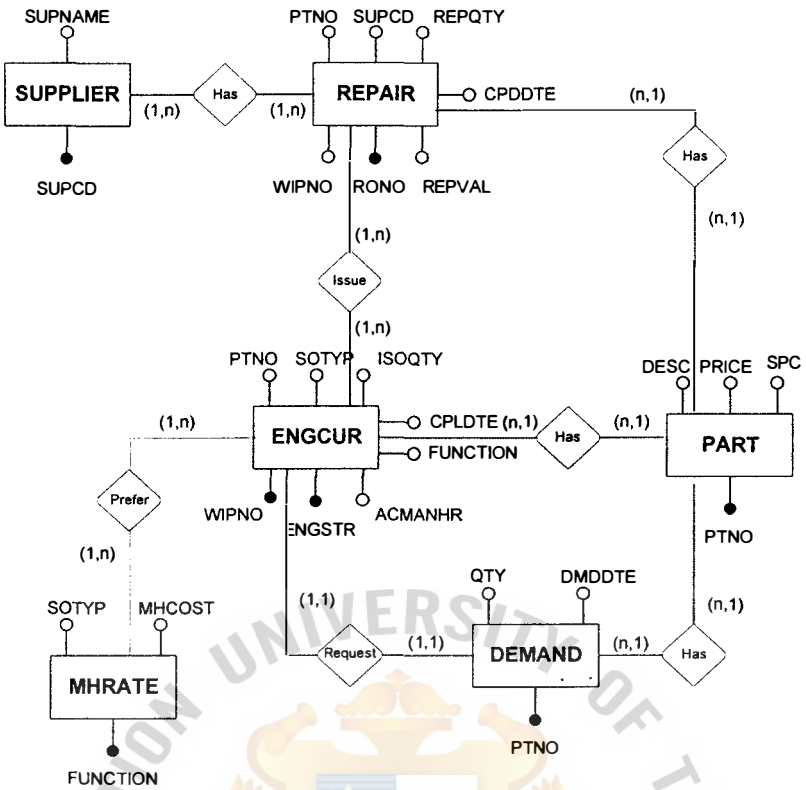


Figure 5.1. The Engine Cost E/R Scheme.

5.3 Definitions of Data Codes

The first step in preparing operational data stores is classifying the valuable attributes of each data entity. The definitions of the data entities follow:

(1) ENGCUR is the data entity that records the current detailed engine maintenance process in a year. The key attributes of engine cost include:

(a) Engine structures code (ENGSTR), which shows the engine serial number that is being reassembled in the maintenance process, the cause of damage, or the end of the cycle hour.

(b) Work-in-Process Number (WIPNO), the job card issued for any maintenance work.

(c) Part Number (PTNO), which records the components of the engine module.

(d) Shop order type (SOTYP), which shows the status code of each part. For instance, code *INK* means the part has been scrapped, while *VEN* means the part has been sent for repair outside the company.

(e) Shop order quantity (ISOQTY), which is the quantity of the reassembled part.

(f) Complete Date (CPLDTE), which is the date of issuing WIPNO.

(g) Function code (FUNCTION), which records the division or department responsible for activating an engine module.

(h) Actual man-hour (ACMAN), which records the lead-time (in hour) activation of the engine module.

(2) REPAIR is the data entity that records data on repairs. It includes;

(a) the part number (PTNO),

(b) the supplier or subcontractor code (SUPCD),

(c) the repair quantity (REPQTY),

(d) the repair values (REPVAL),

(e) the completed date (CPDDTE),

(f) the repair order number (RONO), and

(g) the work-in-progress number (WIPNO).

(3) SUPPLIER is the data entity that stores the supplier code (SUPCD) and supplier name (SUPNAME), which are also present in the warehouse database as a parameter table.

(4) PART is the data entity that stores data on engine spare parts. This includes:

(a) the part number (PTNO),

(b) the description (DESC), and

(c) the price, and

(d) spare part classifications (SPC): SPC1 indicates a consumable part, SPC2 indicates a repairable part, SPC3 indicates a rotable part, SPC5 indicates a Mod Kit, SPC7 indicates a recoverable part, and SPC9 indicates a raw material.

(5) DEMAND is the data entity that stores material requisitions from shop mechanics. This includes:

(a) the part number (PTNO),

(b) the demand date (DMDDTE), and

(c) the demand quantity (QTY).

(6) MHRATE is the data entity that stores the man-hour rate per function. This is used to calculate the man-hour cost per shop visit.

Two main components of data on engine costing are parts and suppliers. Data on engine parts suppliers is indicated by the entity SUPPLIER, while data on engine spare parts is indicated by the entity PART. Other components of engine costing are targets, user requirements, proposals, costs, man-hours, and subcontractors, all of which are recorded as historical data, which also includes the entities ENGCUR, DEMAND, MHRATE, and REPAIR. Figure 5.1 shows in E/R form the data entities and their relationship to each other.

The next step involves data cleansing and transforming it into the warehouse database (Figure 5.2). This is done in four distinct subroutines:

(1) Use details of the ENGCUR entity as base data

(2) Merge related data entities to derive repair-cost and material cost

(3) Store the resulting data into the new entity ENGCOST

(4) Use SQL functions to manipulate associated data to summarize and store into ENGCOST

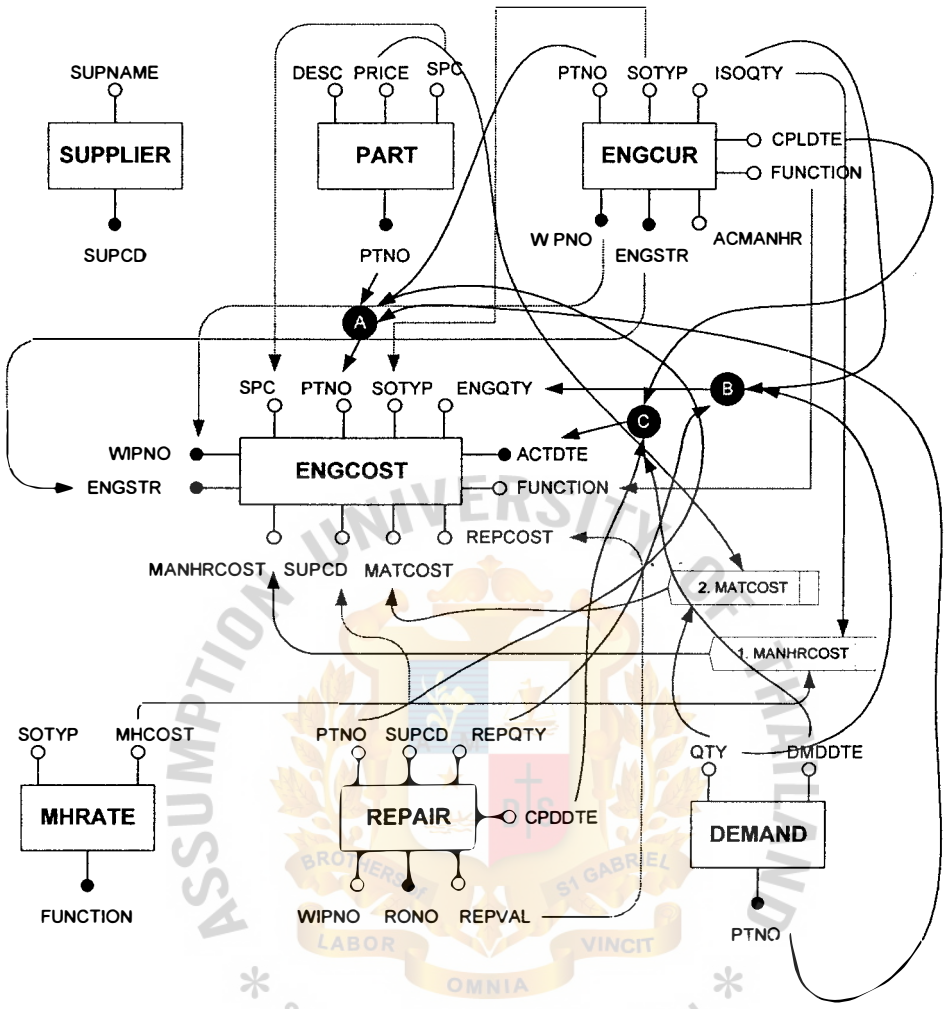


Figure 5.2. Data Cleansing and Transformation.

Data entity ENGCUR is selected as base data because it contains most of the data related to the other data entities (i.e., engine cost information, used parts number, time consumed). All attributes are recorded into the new data entity ENGCOST.

Data resulting from aggregation are also recorded into entity ENGCOST. For example, MANHRCOST comes from calculating ISOQTY multiplied by MHCOST. The program for generating MANHRCOST follows:

```
SELECT A.ENGSTR, A.WIPNO, SUM(A.ISOQTY * B.MHCOST)
FROM   ENGCUR A MHRATE B
ORDER BY A.ENGSTR, A.WIPNO
GROUP BY A.ENGSTR, A.WIPNO
```

The resulting data is stored in the warehouse database, and added to the entities PART and SUPPLIER as parameter table, where it is accessible for future reference.

5.4 A DVR Data Model Example

Building a DVR data model from a warehouse includes classification, partitioning, and clustering reorganize data from relational format into metric or cubic patterns. However, users should first modify or validate the data model before generating the physical structure of multidimensional database. The Visual Basic program is written to solve the methodology of building DVR data model as prototype, the engine cost data are transformed into Microsoft Access'97 database format. This prototype demonstrates any process of building DVR data model at presentation.

The data entities of ENGCOST database include ENGCOST and SUPPLIER (Figure 5.1*)*. The structure of data entities and their characteristics follows in Table 5.1 and Table 5.2:

Table 5.1. ENGCOST Data Entity.

Attribute Name	Data Type	Length
ACTDTE (Action Date) [Index key]	CHAR	7
ENGSTR (Engine Structure Serial Number) [Index key]	CHAR	8
PTNO (Part Number) [Index key]	CHAR	7
SUPCD (Subcontractor Code)	CHAR	5
WIPNO (Work In Process Number) [Index key]	CHAR	4
MANHRCOST (Man-hour Cost)	NUM	12.2
MATCOST (Material Cost)	NUM	12.2
REPCOST (Repaired Cost)	NUM	12.2

Table 5.2. SUPPLIER Data Entity.

Attribute Name	Data Type	Length
SUPCD (Subcontractor Code) [Indexed key]	CHAR	5
SPNAME (Subcontractor Name)	CHAR	30

5.4.1 Example of DVR data model generation

The steps of generating DVR data model from engine cost database into multidimensional form shown in Figure 3.2 are composed of classification, partitioning, and clustering module. This example reviews any processes and steps of activities to build DVR data model, and finally solves the statement of the problems of this research.

(1) Data Classification

The classification module indicates the candidate dimensions or candidate variables from the attributes of warehouse databases using data type or characteristics of any attributes.

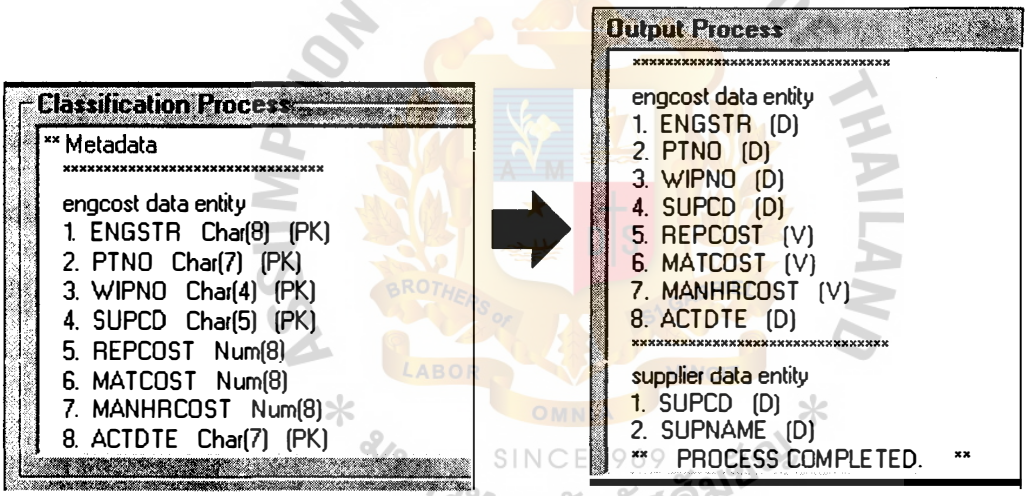


Figure 5.3. Example of Classification Process in Program prototype.

Figure 5.3 shows input and output screen of classification process using Visual Basic Program as prototype to indicate candidate dimension and variable. When the fundamental characteristics of attribute ENGSTR, PTNO, WIPNO, SUPCD, and ACTDTE in ENGCOST data entity are presented in the index-key fields, these characteristics are defined as candidate dimensions (show indicator 'D' in Figure 5.3). The characteristic of attribute SUPCD of SUPPLIER data entity also presented as index-key field will be defined as candidate dimension.

The data type of attribute SNAME in SUPPLIER data entity is presented as Character format will be defined as candidate dimension. On the other hand, the data types of attributes REPCOST, MATCOST, and MANHRCOST of ENGCOST data entity are numeric, they are defined as candidate variables (show indicator 'V' in Figure 5.3).

(2) Data Partition

The partitioning module provide the two steps: data mapping and Internal Comparison Process (ICP) to assign the actual dimension, variable, and relative dimension (DVR) and transform data set into DVR data model.

In the mapping process, the candidate dimensions of ENGSTR, PTNO, WIPNO, SUPCD, and ACTDTE are defined as base attributes and map with other attributes of ENGCOST data entity. The mapping data sets, yield from data mapping, are then manipulated using ICP algorithms in the next process.

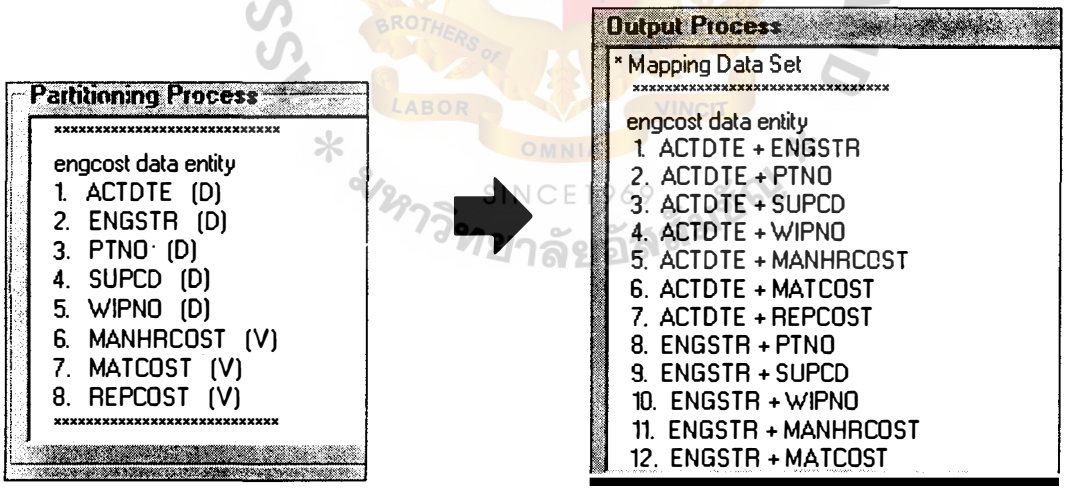


Figure 5.4. Example of Partition Process in Program prototype.

Figure 5.4 shows the mapping data sets of ENGCOST data entity. The candidate dimension ACTDTE is assigned to be base and mapped with other attributes, the mapping data sets are shown as follows:

ACTDTE + ENGSTR, ACTDTE + PTNO, ACTDTE + SUPCD, ACTDTE + WIPNO,
ACTDTE + MANHRCOST, ACTDTE + MATCOST, ACTDTE + REPCOST,

ACTDTE + ACTDTE, ACTDTE + ENGSTR + PTNO, ACTDTE + ENGSTR + SUPCD, ACTDTE + ENGSTR + WIPNO, ACTDTE + ENGSTR + MANHRCOST, ACTDTE + ENGSTR + MATCOST, ACTDTE + ENGSTR + REPCOST, ACTDTE + ENGSTR + PTNO + SUPCD, ACTDTE + ENGSTR + PTNO + WIPNO, ACTDTE + ENGSTR + PTNO + MANHRCOST, ACTDTE + ENGSTR + PTNO + MATCOST, ACTDTE + ENGSTR + PTNO + REPCOST, ACTDTE + ENGSTR + SUPCD + WIPNO, ACTDTE + ENGSTR + SUPCD + MANHRCOST, ACTDTE + ENGSTR + SUPCD + MATCOST, ACTDTE + ENGSTR + SUPCD + REPCOST, ACTDTE + ENGSTR + WIPNO + MANHRCOST, ACTDTE + ENGSTR + WIPNO + MATCOST, ACTDTE + ENGSTR + WIPNO + REPCOST, ACTDTE + ENGSTR + MANHRCOST + MATCOST, ACTDTE + ENGSTR + MANHRCOST + REPCOST, ACTDTE + ENGSTR + MATCOST + REPCOST, ACTDTE + ENGSTR + PTNO + SUPCD + WIPNO, ACTDTE + ENGSTR + PTNO + SUPCD + MANHRCOST, ACTDTE + ENGSTR + PTNO + SUPCD + MATCOST, ACTDTE + ENGSTR + PTNO + SUPCD + REPCOST, ACTDTE + ENGSTR + PTNO + SUPCD + WIPNO + MANHRCOST, ACTDTE + ENGSTR + PTNO + SUPCD + WIPNO + MATCOST, ACTDTE + ENGSTR + PTNO + SUPCD + WIPNO + REPCOST, ACTDTE + ENGSTR + PTNO + SUPCD + MANHRCOST + MATCOST, ACTDTE + ENGSTR + PTNO + SUPCD + MANHRCOST + REPCOST, ACTDTE + ENGSTR + PTNO + SUPCD + MANHRCOST + MATCOST + REPCOST

The other candidate dimensions are also assigned to be bases and mapped with attributes of data entity. There are 170 mapping data sets resulted from engine cost database example.

The mapping data sets are prepared for access in ICP to validate the actual dimension, variable, and relative dimension. Process 1 of ICP selects only one of the mapping data sets that are similar, and deletes the rest. For example, mapping data set of ACTDTE + ENGSTR and ENGSTR + ACTDTE, process 1 will select only one mapping data set (ACTDTE + ENGSTR) and delete the mapping data set (ENGSTR + ACTDTE).

Process 2 deletes the mapping data sets that candidate variables are assigned to be base attributes, e.g. MANHRCOST + ACTDTE, MATCOST + ACTDTE, or REPCOST + ACTDTE, etc.

Process 3 and Process 4 indicate the relation attributes of mapping data sets. The mapping data sets (ACTDTE + ENGSTR, ACTDTE + PTNO, ACTDTE + SUPCD, ACTDTE + WIPNO, ACTDTE + MANHRCOST, ACTDTE + MATCOST, ACTDTE + REPCOST) of ENGCOST data entity undergo Process3, the comparison of attributes use the SQL query process to count the value items and inspect the relation. The solution of candidate dimensions shows the different

count value items and all of them are index-key fields, so the process automatically represents them as actual dimensions ($D_1.ACTDTE$, $D_2.ENGSTR$, $D_3.PTNO$, $D_4.SUPCD$, $D_5.WIPNO$). While the solution of candidate variables show the different count value items that may be related to each candidate dimensions, the process also represents them as actual variables ($V_1.MANHRCOST$, $V_2.MATCOST$, $V_3.REPCOST$). On the other hand, the mapping data sets (SUPCD + SUPNAME) of SUPPLIER data entity undergo process 3, the count value items of SUPCD and SUPNAME are similar. The data type of SUPCD is index-key field, then attribute SUPCD is assigned to be actual dimension, and attribute SUPNAME is assigned to be relative dimension. Process 4 presents the relationship of this mapping data set as $D_4.SUPCD \leftrightarrow R_1.SUPNAME$.

The mapping data set (SUPNAME + SUPCD) is erased in Process 5 because the attribute SUPNAME is assigned to be relative dimension. Process 6 replaces the joint function symbol of mapping data sets to be associated function symbol. Process 7 deletes the duplication of mapping data set, e.g. data sets ($D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$; $D_1.ACTDTE \Rightarrow V_1.MANHRCOST$; $D_1.ACTDTE \Rightarrow V_2.MATCOST$; $D_1.ACTDTE \Rightarrow V_3.REPCOST$), the mapping data sets ($D_1.ACTDTE \Rightarrow V_1.MANHRCOST$; $D_1.ACTDTE \Rightarrow V_2.MATCOST$; $D_1.ACTDTE \Rightarrow V_3.REPCOST$) are deleted.

Process 8 replaces the mapping data sets into DVR data models, the outcome of this process results in the 32 mapping data sets. The DVR data models are shown as follows:

- $D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

- $D_1.ACTDTE + D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_4.SUPCD \rightarrow D_8.SUPNAME$

(3) Data Clustering

The last step of clustering process rearranges the 32 mapping data sets of partitioning process, and performs the mapping table. The first mapping data set is assigned to be base data set and mapped with other mapping data sets (see Table 5.3). The mapping processes are activated until the last of the mapping data set is assigned to be base data set or related data set is empty.

Table 5.3. Example Data Mapping of Clustering Process.

Base Data Set: $D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

Related Data Set	Grouping Relation Set
$D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

Table 5.3. Example Data Mapping of Clustering Process (continued).

Base Data Set: $D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

Related Data Set	Grouping Relation Set
$D_1.ACTDTE + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_2.ENGSTR + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_2.ENGSTR + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_2.ENGSTR + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_2.ENGSTR + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

Table 5.3. Example Data Mapping of Clustering Process (continued).

Base Data Set: $D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

Related Data Set	Grouping Relation Set
$D_2.ENGSTR + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_2.ENGSTR + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_2.ENGSTR + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_2.ENGSTR + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_1.ACTDTE + D_2.ENGSTR + D_3.PTNO + D_4.SUPCD + D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
$D_4.SUPCD \rightarrow D_8.SUPNAME$	$D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST \cup D_4.SUPCD \rightarrow D_8.SUPNAME$

There are 496 grouping relation sets from the mapping process, the grouping relation sets are then manipulated in External Comparison Process (ECP) algorithms to form new data sets. All grouping relation sets mainly come from ENGCOST data entity, so there are no more relations occurring in this engine cost example, the final DVR data models are composed of:

- $D_1.ACTDTE \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_5.WIPNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_2.ENGSTR \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_3.PTNO \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$
- $D_1.ACTDTE + D_4.SUPCD \Rightarrow V_1.MANHRCOST, V_2.MATCOST, V_3.REPCOST$

- D_1 ACTDTE + D_2 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 PTNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 SUPCD ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 PTNO + D_2 SUPCD ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 PTNO + D_2 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 SUPCD + D_2 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 PTNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 SUPCD ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 PTNO + D_3 SUPCD ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 PTNO + D_3 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 SUPCD + D_3 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 PTNO + D_3 SUPCD ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 PTNO + D_3 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 SUPCD + D_3 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 PTNO + D_2 SUPCD + D_3 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 PTNO + D_4 SUPCD ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 PTNO + D_4 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 SUPCD + D_4 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 PTNO + D_3 SUPCD + D_4 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ENGSTR + D_2 PTNO + D_3 SUPCD + D_4 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 ACTDTE + D_2 ENGSTR + D_3 PTNO + D_4 SUPCD + D_5 WIPNO ⇒ V_1 MANHRCOST, V_2 MATCOST, V_3 REPCOST
- D_1 SUPCD → D_2 SUPNAME

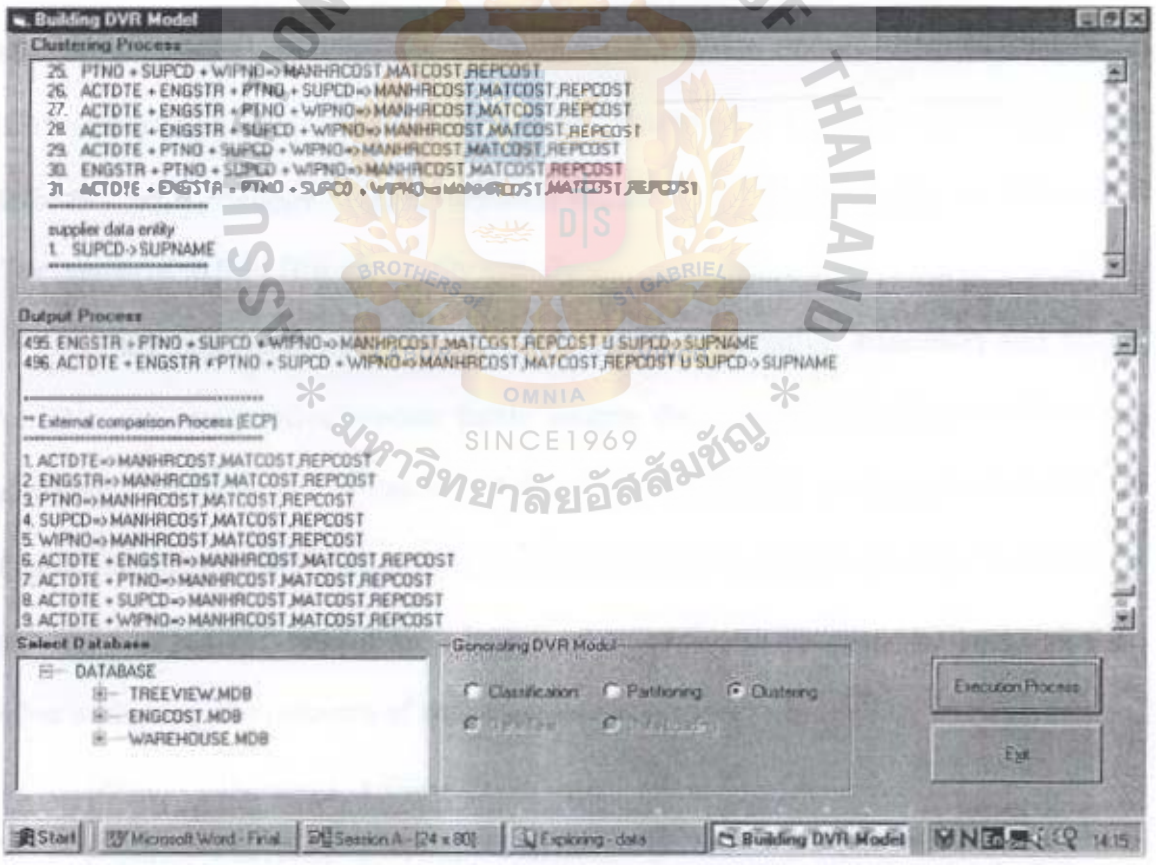


Figure 5.5. Example of Clustering Process.

Figure 5.5 shows the example of clustering process using two steps: mapping process and ECP execution. The output of ECP execution lists the DVR data models of engine cost example. This concludes the procedural descriptions for building a DVR data model from a warehouse database using three processes of classification, partitioning, and clustering reorganize data from relational format into metric or cubic patterns. The next step is transforming the DVR data model into physical data structure of multidimensional database and using SQL commands to load data into physical structure.

5.4.2 DVR Data Model Design via Graph Model

In designing DVR data model of graph representation, any attributes of warehouse database are defined to be candidate dimension or variable in the same way as classification process (see Figure 5.3). The results of ENGCOST data entity are as follows: D_{PTNO}, D_{ENGSTR}, D_{SUPCD}, D_{WIPNO}, D_{ACTDTE}, $V_{MANHRCOST}$, $V_{MATCOST}$, $V_{REPCOST}$. The results of SUPPLIER data entity are: D_{SUPCD}, $D_{SUPNAME}$.

In the process of identifying actual dimension, relative dimension and actual variable, the comparative process firstly assigns the primary index key as a base for mapping. The indexed key fields of ENGCOST data entity are composed of PTNO, ENGSTR, SUPCD, WIPNO, and ACTDTE attributes. Assigning the PTNO attribute as primary indexed key field, PTNO is defined to be based attributes and mapped with other attributes. The outcome of data mapping is:

$\{D_{PTNO} \cup D_{ENGSTR}\}$, $\{D_{PTNO} \cup D_{SUPCD}\}$, $\{D_{PTNO} \cup D_{WIPNO}\}$,
$\{D_{PTNO} \cup D_{ACTDTE}\}$, $\{D_{PTNO} \cup V_{MANHRCOST}\}$,
$\{D_{PTNO} \cup V_{MATCOST}\}$, $\{D_{PTNO} \cup V_{REPCOST}\}$

Use a comparative algorithm to screen the data type and relation and assign to be actual dimension, relative dimension, and actual variable. This step reveals that the number of unique items in dimension PTNO is different from the other attributes. This

proves that dimension PTNO is not dependent on other attributes, then assigned dimension ENGSTR, SUPCD, WIPNO, and ACTDTE are actual dimensions, and assigned variable MANHRCOST, MATCOST, and REPCOST are actual variables. The SUPPLIER data entity is then executed using comparative algorithm. This step proves that SUPCD has a relationship with SUPNAME due to the similarity of the count value items. The dimension SUPNAME is assigned to be relative dimension.

The following graph presentation shown in Figure 5.6 introduces a DVR data model of ENGCOST and SUPPLIER data entities.

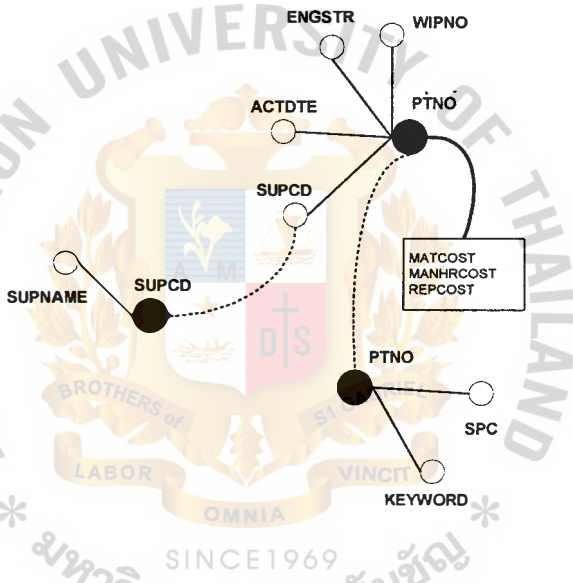


Figure 5.6. Engine Cost Database in 2D Conceptual Model.

The indexed keys of ENGCOST data entity comprise the ENGSTR, PTNO, SUPCD, WIPNO, and ACTDTE attributes. The PTNO attribute is selected as the main dimension and the other candidate dimension is attributed as sub-dimensions. The resulting graphic representation is shown in Figure 5.6. The SUPPLIER data entity connects the ENGCOST data entity's attributes, which show the relationship between data entities. Other ways, such as time series in drill down or drill up, can show other perspectives in the multidimensional cubes.

5.5 Conclusion

This chapter uses the methodologies described in Chapters 3 and 4 on an airline's database related to engine cost in particular. In doing so, the data model and graph model formats are illustrated, as well as the process of generating a DVR data model. The process includes classification module, partitioning module, clustering module and graph model applications to automatically generate the DVR data model. Finally, these models are tested by transforming them into an OLAP software package (Oracle Personal Express) to view screen layout patterns.

VI. CONCLUSIONS AND RECOMMENDATIONS

This research is to present a process that automatically generates a multidimensional data structure called DVR, useful in improving storage and analysis of great amounts of corporate data. The process transforms the current way of storing flat data in relational form in a warehouse database, so that it is re-stored in a form where all its possible relationships to other data are interconnected. This allows the most creative analytical queries even in huge masses of stored data.

The research presents a design process for a multidimensional data storage structure shown in Figure 6.1. To test the program algorithms, a prototype was used: engine cost in an airline.

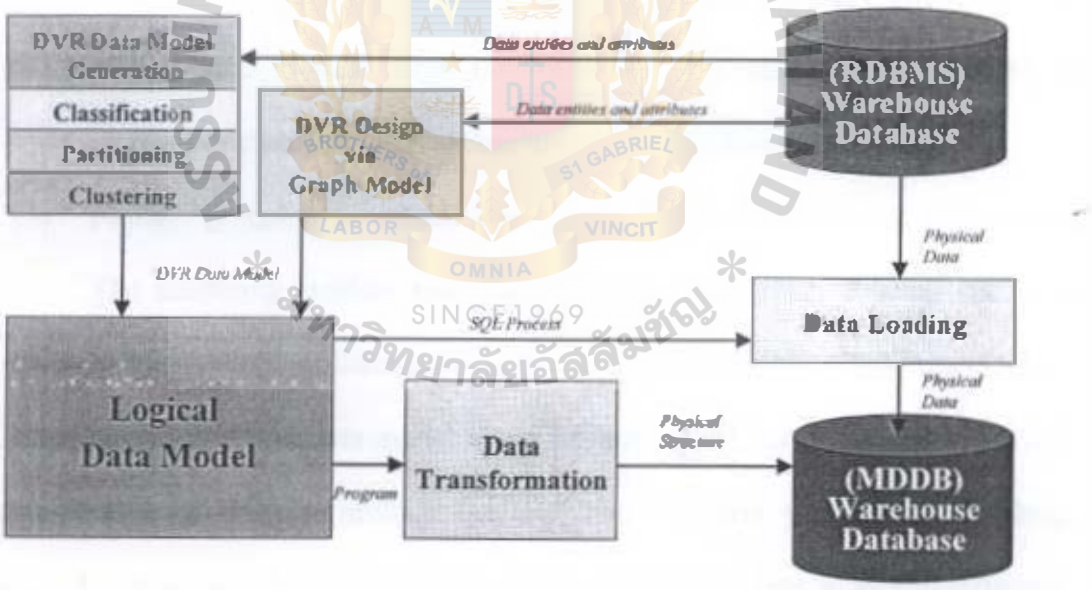


Figure 6.1. Data Access of DVR data model.

The research also presents a method of automating the process of generating multidimensional relationships patterns in a database. This process yields more complex patterns as well as screens out irrelevant data. It also minimizes human input and human errors in the processing of voluminous amounts of data. These objectives are all

accomplished. The research successfully submits a procedure for generating logical data structures automatically, the design for the process, and the input-output schema for multidimensional data storage structure design.

In the DVR data model generation, building a DVR data model via methodology of classification, partitioning, and clustering procedure first clears all elements in the data entities, transforms them into a multidimensional format, builds a model of the multidimensional structure, and then loads the data into the physical multidimensional structure. These procedures provide:

(1) The classification module screens, identifies, and labels each attribute in the data entity as either a candidate dimension or a variable of hypercubes.

(2) Formation of mapped data sets, the partitioning module selects a candidate dimension as a base for mapping with the other attributes inside the data entity. This module uses the Internal Comparison Process (ICP) to manipulate data sets that validate the multidimensional format.

(3) Finally, grouping of related dimensions and their associated data values. The clustering module uses the External Comparison Process (ECP) to manipulate all multidimensional data structures in the partitioning module.

The design of a DVR data model using the graph model approach provides data support intelligence through multidimensional data analysis when there are huge amounts of online data by transforming a data warehouse into a multidimensional storage format.

The design approach uses a graph model data catalog that generates the DVR structure in the form of a multidimensional cube, called a hypercube. The graph structure's center node represents the fastest varying dimension, several branch nodes represent associate dimension attributes, and rectangles represent index data values and

measurements. These may also be represented on fact schemes that integrate relative information and external sources, as well as extend the algorithms to build DVR patterns in standard SQL form.

The prototype developed in this research shows how workload is reduced and details the input-output design phases of a Multidimensional Database (MDDB). This model is created with this four-point focus: (1) to create a DVR that meets user requirements more accurately, (2) to design a routine that automatically creates an MDDB database format, (3) to speed up the input-output design phase, and (4) to minimize human error. The advantage of the two approaches solves the statement of problems as follows:

(1) Ease of handling large database.

Data volume is often the first issue that comes to mind when evaluating warehouse cost and benefit. The handling of huge amount of data in the warehouse database is a big task. So the automatic generating data model may reduce the assignment of the designing phase, making a greater and faster progress in solving the volume problem, and perform loading and querying process that is affordable and flexible. It also provides ease-of-use of the data model and query tools. Users are able to check the error of data from data model that consists of missing data, data characteristics, and their relations. For example, the distinction between relative attributes is the same as Supplier Code and Supplier Name. The data model shows the different data volumes of Supplier Code compared with Supplier Name. Users check the error occurred in the warehouse database and correct them.

(2) Reduce the technical expertise required in the designing phase.

The system automatically generates the DVR data model supporting user requirements and also reduces the technical skill of analyst in the designing phase. The DVR data model shows the easiest way to manipulate and transform the physical data structure in pattern of hypercube form. End-users well understand the pattern of analysis format in graph model before loading data and generating the physical structure of multidimensional database. Any mistaken forms are corrected manually. These processes will also reduce the cost and the time of the designing phase.

(3) Support user requirement.

The automatic generating DVR data model procedures are developed to manipulate the current detailed data level of warehouse database, which store the significant data required from users, into patterns of multidimensional forms. These procedures identify the valuable data in the frame of dimensional model that shows the logical grouping relation of dimensions, fact or measures, and the dimensional matrices. The output data model comes from the original source of warehouse database, analysts are not certain to design the hypercube patterns with learning the characteristic and relation of any attributes and data entities in relational warehouse databases. The DVR data model may be served the effective and efficient DSS process to responsive managers.

The designing processes of DVR data model of two approaches are developed to prove DVR data model by using engine cost example. The research also focuses on the logical data structure in the form of hypercube pattern, the data transformation and data loading are preferred to any type of databases (e.g. Oracle DBMS, IBM OS/2, Informix

DBMS, Sybase DBMS, etc.) that require computer programs or JCL programs in data access layer. Further task requires applying DVR data model in the form of star-schema, snowflake, and other newest concepts to serve DSS functions. Finally, the additional function that requires more DVR data model is generating hierarchical level of dimensions. The relation of attributes may be attached to the same dimension. For example, Part Number and Spare Part Class attributes may be set to be the level of PART dimension. The high level of PART dimension shows the spare part class of part information, the next level of each spare part class shows the part numbers.

DATA TRANSFORMATION AND DATA LOADING

This session presents the transformation between logical data model and physical data structure of multidimensional database. The majority of research shows the processes of building DVR model (see Figure 6.1), however the next steps of transformation are summarized in appendix A using data model of Example 3.

For building physical data model, the system will automatically replace logical DVR model with the pattern of multidimensional database related to OLAP applications. This research uses Oracle Personal Express tools as prototype to substitute data model into physical data model. The data model of example 3 will be replaced as follows:

(1) Dimension

The system automatically generates the SCODE, ACODE, CCODE, and PCODE dimensions in the form of express command using descriptive data in data directory or metadata such as length of data, data description etc. Data are then loaded from relational database using SQL process into multidimensional database.

Dimensions: $D_1^1.SCODE$, $D_2^1.ACODE$, $D_1^2.CCODE$, $D_2^2.PCODE$

Personal Express Command:

DEFINE SCODE DIMENSION TEXT WIDTH 5
LD Salesman Code
DEFINE ACODE DIMENSION TEXT WIDTH 3
LD Area Code
DEFINE CCODE DIMENSION TEXT WIDTH 6
LD Customer Code
DEFINE PCODE DIMENSION TEXT WIDTH 5
LD Product Code

(2) Variable

After dimensions are already created, the next proceeding automatically generates the SALEVAL, COMM, and COSTVAL variables as follows:

Variables or Measures: $V_1^2.SALEVAL$, $V_2^1.COMM$, $V_2^2.COSTVAL$

Relationship: $D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

DEFINE SALEVAL VARIABLE <CCODE PCODE SCODE ACODE>
LD Sale Values Information

DEFINE COMM VARIABLE <CCODE PCODE SCODE ACODE>
LD Commission Values Information

DEFINE COSTVAL VARIABLE <CCODE PCODE SCODE ACODE>
LD Cost Values Information

(3) Relative Dimension

The relative dimension of SNAME, and PNAME attributes are continuously created as follows:

Relative Dimensions: $R_1^1.SNAME, R_1^2.PNAME$

Relationship: $D_1^1.SCODE \leftrightarrow R_1^1.SNAME, D_2^2.PCODE \leftrightarrow R_1^2.PNAME$

Personal Express Command:

DEFINE SNAME VARIABLE TEXT <SCODE>
LD Salesman Name
DEFINE PNAME VARIABLE TEXT <PCODE>
LD Product Name

(4) Data Loading

To load the large volume of data from relational database into physical structure of multidimensional data model, the above data models of example 3 will be generated into *SQL* statement as follows:

Data Model: $D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$

SQL Process:
Select *SCODE*, sum(*SALEVAL*), sum(*COMM*)
From *SALE*
Group by *SCODE*
Order by *SCODE*

Data Model: $D_1^1.SCODE \leftrightarrow R_1^1.SNAME$

SQL Process:
Select *SCODE*, *SNAME*
From *SALE*
Order by *SCODE*

Data Model: $D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM$

SQL Process:
Select *ACODE*, sum(*SALEVAL*), sum(*COMM*)
From *SALE*
Group by *ACODE*
Order by *ACODE*

Data Model: $D_1^1.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM$

SQL Process:
Select *SCODE*, *ACODE*, sum(*SALEVAL*), sum(*COMM*)
From *SALE*
Group by *SCODE, ACODE*
Order by *SCODE, ACODE*

Data Model: $D_1^2.CCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$

SQL Process:
Select *CCODE*, sum(*SALEVAL*), sum(*COSTVAL*)
From *COST*
Group by *CCODE*
Order by *CCODE*

Data Model: $D_2^2.PCODE \leftrightarrow R_1^2.PNAME$

SQL Process:
Select *PCODE, PNAME*
From *COST*
Group by *PCODE*

Data Model: $D_2^2.PCODE \Rightarrow V_1^2.SALEVAL\ ,\ V_2^2.COSTVAL$

SQL Process:
Select *PCODE*, sum(*SALEVAL*), sum(*COSTVAL*)
From *COST*
Group by *PCODE*

Order by *PCODE*

Data Model: $D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$

SQL Process: Select *SCODE*, sum(*SALEVAL*), sum(*COSTVAL*)
From *COST*
Group by *SCODE*
Order by *SCODE*

Data Model: $D_1^2.CCODE \infty D_2^2.PCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$

SQL Process: Select *CCODE, PCODE*, sum(*SALEVAL*), sum(*COSTVAL*)
From *COST*
Group by *CCODE, PCODE*
Order by *CCODE, PCODE*

Data Model: $D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$

SQL Process: Select *CCODE, SCODE*, sum(*SALEVAL*), sum(*COSTVAL*)
From *COST*
Group by *CCODE, SCODE*
Order by *CCODE, SCODE*

Data Model: $D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,\ V_2^2.COSTVAL$

SQL Process: Select *PCODE, SCODE*, sum(*SALEVAL*), sum(*COSTVAL*)
From *COST*
Group by *PCODE, SCODE*
Order by *PCODE, SCODE*

Data Model: $D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \Rightarrow V_1^2.SALEVAL,$

$V_2^2.COSTVAL$

SQL Process: Select *CCODE, PCODE, SCODE*, sum(*SALEVAL*),
sum(*COSTVAL*)
From *COST*
Group by *CCODE, PCODE, SCODE*
Order by *CCODE, PCODE, SCODE*

Data Model: $D_1^1.SCODE \Rightarrow V_1^1.SALEVAL,\ V_2^1.COMM,\ V_2^2.COSTVAL$

SQL Process: Select *A.SCODE*, sum(*A.SALEVAL*), sum(*B.COMM*),
sum(*B.COSTVAL*)

From *SALE A, COST B*
Where *A.SCODE = B.SCODE*
Group by *A.SCODE*
Order by *A.SCODE*

Data Model: $D_1^2.CCODE \infty D_3^2.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM,$

$V_2^2.COSTVAL$

SQL Process: Select *A.CCODE, A.SCODE*, sum(*B.SALEVAL*), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.CCODE, A.SCODE*
Order by *A.CCODE, A.SCODE*

Data Model: $D_2^2.PCODE \infty D_1^1.SCODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM,$

$V_2^2.COSTVAL$

SQL Process: Select *A.PCODE, B.SCODE*, sum(*B.SALEVAL*), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.PCODE, B.SCODE*
Order by *A.PCODE, B.SCODE*

Data Model: $D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^1.SALEVAL, V_2^1.COMM,$

$V_2^2.COSTVAL$

SQL Process: Select *A.SCODE, B.ACODE*, sum(*B.SALEVAL*), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.SCODE, B.ACODE*
Order by *A.SCODE, B.ACODE*

Data Model: $D_1^2.CCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL,$

$V_2^1.COMM, V_2^2.COSTVAL$

SQL Process: Select *A.CCODE, A.SCODE, B.ACODE*, sum(*A.SA*LEVAL), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.CCODE, A.SCODE, B.ACODE*

Order by *A.CCODE, A.SCODE, B.ACODE*

Data Model: $D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow V_1^2.SALEVAL,$

$V_2^1.COMM, V_2^2.COSTVAL$

SQL Process: Select *A.PCODE, A.SCODE, B.ACODE*, sum(*A.SALEVAL*), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.PCODE, A.SCODE, B.ACODE*
Order by *A.PCODE, A.SCODE, B.ACODE*

Data Model: $D_1^2.CCODE \infty D_2^2.PCODE \infty D_3^2.SCODE \infty D_2^1.ACODE \Rightarrow$

$V_1^2.SALEVAL, V_2^1.COMM, V_2^2.COSTVAL$

SQL Process: Select *A.CCODE, A.PCODE, A.SCODE, B.ACODE*, sum(*A.SALEVAL*), sum(*B.COMM*), sum(*A.COSTVAL*)
From *COST A, SALE B*
Where *A.SCODE = B.SCODE*
Group by *A.CCODE, A.PCODE, A,SCODE, B.ACODE*
Order by *A.CCODE, A.PCODE, A,SCODE, B.ACODE*

BIBLIOGRAPHY

1. A. H. M. ter Hofstede, H. A. Proper, Th. P. van der Weide, "Data Modeling in Complex Application Domains", CaiSE, Manchester, UK, pp. 364-377, 1992.

2. Aberer, Karl & Klemens Hemm, "A Methodology for Building a Data Warehouse in a Scientific Environment", pp.90-101, 1996.

3. Abrial J.R., "Data Semantics Database Management", Klimnie & Koffinan eds., North-Holland, 1974.

4. Adriaans, Pieter & Dolf Zantinge, Data Mining, Addison-Wesley Publishing, 1996.

5. Agarwal, S. & Rakesh Agrawal, Prasad Deshpande, Ashish Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, Sunita Sarawagi, "On the Computation of Multidimensional Aggregates", VLDB, Bombay, India, 1996.

6. Agosti, Maristella & Robert Colotti, Girolamo Gradenigo, "A Two-Level Hypertext Retrieval Model for Legal Data", SIGIR, pp. 316-325, 1991.

7. Agrawal, Divyakant & Amr El Abbadi, Ambuj K. Singh, Tolga Yurek, "Efficient View Maintenance at Data Warehouses", SIGMOD Conference, Tucson, Arizona, pp.417-427, 1997.

8. Akinde, Micheal O. & Ole Guttorm Jensen, Michael H. Bohlen, "Minimizing Detail Data in Data Warehouses", EDBT, Valencia, Spain, pp.293-307, 1998.

9. Alalouf, Carole Hybrid OLAP: The best of both worlds, A white paper by speedware corporation, 1997.

10. Albano A., Orsini R., "A Software Engineering Approach for Database Design: The GALILEO Project, Computer Aided Database Design", North-Holland, 1985.

11. Albrecht, Jens & Wolfgang Sporer, "Aggregate-Based Query Processing in a Parallel Data Warehouse Server", DEXA Workshop, Florence, Italy, pp.40-44, 1999.

12. Arbor Software, "The Role of the Multidimensional Database in a Data Warehousing Solution".

13. Astrahan, Morton M. & Edward B. Altman, P. L. Fehder, Michael E. Senko, "Concepts of a Data Independent Accessing Model", SIGFIDET Workshop, pp. 349-382, 1972.

14. Atkinson M. & Chisholm K., Cockshott P., "PS-ALGOL: An ALGOL with a Persistent Heap", SIGPLAN Notices, 17(7), 1982.

15. Bachman, Charles W. & Manilal Daya, "The Role Concept in Data Models", VLDB, Tokyo, Japan, pp.464-476, 1977.

16. Baekgaard, Lars "Event-Entity-Relationship Modeling in Data Warehouse Environments", DOLAP, Kansas City, Missouri, USA, pp.9-14, 1999.

17. Ballard, Chuck & Dirk Herreman, Don Schau, Rhonda Bell, "Data Modeling Techniques for Data Warehousing", International Technical Support Organization of IBM, February 1998.

18. Barquin, Ramon & Herb Edelstein, Building, Using, and Managing the Data Warehouse, The Data Warehousing Institute Series from Prentice Hall PTR, New Jersey, 1997.

19. Batini C. & G.D. Battista, "A Methodology for Conceptual Documentation and Maintenance", Information Systems 13(3), pp.297-318, 1988.

20. Bischoff, Joyce & Ted Alexander, Data Warehouse: Practical Advice from the Experts, Prentice Hall, New Jersey, 1997.

21. Bosworth, Adam & Andrew Layman, Hamid Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total", SIGMOD, Seattle, Washington, 1998.

22. Boufaida, Mahmoud & Zizette Boufriche-Boufaida, "On Extending a Semantic Data Model with some Aspects of Rules and Objects.", KRDB, Seattle, Washington, USA, pp. 5.1-5.7, 1998.

23. Bouzeghoub M. "Using Expert Systems in Schema Design", Conceptual Modeling, Database and Case, N.Y., 1992.

24. Bouzeghoub M. & G. Gardarin, et al, "Database Design Tools – An Expert System Approach", Very Large Data Base Conference, Stockholm, 1985.

25. Bouzeghoub M. & I. Comyn-Wattian, "View Integration by Semantic Unification and Transformation of Data Structures", ER Approach, Elsevier Science Publisher, 1991.

26. Buchmann, Dayal U. & McCarthy D., "Rules are Object too: A Knowledge Model for an Active, Object-Oriented Database Management System", Proc. 2nd International Workshop on Object-Oriented Database System, 1988.

27. Buneman O.P., Frankel R.E., "FQL-A Functional Query Language", ACM SIGMOD, Boston, May 1979.

28. Cardenas, Alfonso F. & James P. Sagamang, "Modeling and Analysis of Data Base Organization. The doubly chain tree structure", IS, pp.57-67, 1975.

29. Carkenord, Barbara A. "Why Build A Logical Data Model", Embarcadero Technologies, Inc. San Francisco.

30. Catarci, Tiziana & Giovanna D'Angiolini, Maurizio Talamo, "Conceptual Language for Statistical Data Modeling", DKE 17, pp.93-125, 1995.

31. Chan C-Y, Ioannidis Y., "Hierarchical Prefix Cubes for Range-Sum Queries", 25th International Conference on Very Large Data Bases, Scotland, September 1999.

32. Chaudhuri, Surajit & Umeshwar Dayal, "Data Warehousing and OLAP for decision support", DOOD, Montreux, Switzerland, pp.33-34, 1997.

33. Chen P.P., "The Entity-Relationship Model – Toward a Unified View of Data", ACM Transactions on Database Systems, Vol.1, No.1, pp.9-38, 1976.

34. Chen, I-Min A. & Victor M. Markowitz, "Modeling Scientific Experiments with an Object Data Model", ICDE, Taipei, Taiwan, pp.391-400, 1995.

35. Chen, Peter P. " The Entity-Relationship Model: Toward a Unfied View of Data", VLDB, Framingham, Massachusetts, pp.173, 1975.

36. Choobineh J. & M. Mannino, et al, "An Expert Database Design System Based on Analysis of Forms", pp.242-253, 1988.

37. Codd E. F. and S.B. Codd, OLAP with TM/1 E. F. Codd & Associates, On-Line Analytical Processing (OLAP) White Paper, 1994.

38. Codd E.F., "Extending the Database Relational Model to Capture More Meaning", ACM TODS, Arizona, USA, Vol.4, pp.397-434, 1979.

39. Colby, Latha S. & Richard L. Cole, Edward Haslam, Nasi Jazayeri, Galt Johnson, William J. McKenna, Lee Schumacher, David Wilhite, "Redbrick Vista: Aggregate Computation and Management", ICDE, Orando, 1998.

40. Cooper, Richard & Zhenzhou Qin, "A Graphical Data Modeling Program with Constaint Specification and Management.", BNCOD, Aberdeen, Scotland, pp.192-208, 1992.

41. Covvet C. & C. Proix, et al, "ALECSI: An Expert System for Requirements Engineering", CaiSE'91, Norways, 1991.

42. Cui, Y. and J. Widom, "Lineage Tracing in a Data Warehousing System", Proceedings of the Sixteenth International Conference on Data Engineering, San Diego, California, Feb. 2000.

43. Cui, Y. J. Widom, and J. L. Wiener, "Practical Lineage Tracing in Data Warehouses", Proceedings of the 16th International Conference on Data Engieering, San Diego, California, February 2000.

44. Cui, Y. J. Widom, and J.L. Wiener, "Tracing the Lineage of View Data in a Data Warehousing Environment", Technical Report, Standford University, Stanford, CA, 1997.

45. Czuchry A.J. & D.R. Harris, "A New Paradigm for Requirement Engineering", IEEE Expert, Winter, pp.21-34, 1988.

46. De Antonellis V. & Zonta B., "A Tool for Modeling Dynamics in Conceptual Design", In Computer Aided Database Design, A. Albano et al., North Holland, 1987.

47. De Troyer, Olga & Rene Janssen, "On Modularity for Conceptual Data Models and the Consequences for Subtyping, Inheritance & Overriding", ICDE, Vienna, Austria, pp. 678-685, 1993.

48. Delobel, Claude "Data Base Theory and Modeling – Theoretical and Practical Aspects", VLDB, West Berlin, Germany, pp.112, 1978.

49. Dubois, Eric & Jacques Hagelstein, Eugene Lahou, Andre Rifaut, Fiona Williams, "A Data Model for Requirements Analysis", ICDE, New Orleans, Louisiana, pp. 646-653, 1986.

50. Duong, Toncan & John Hiller, Uma Srinivasan, "A Unifying Model of Data, Metadata and Context", DEXA, Prague, Czech Republic, pp. 68-79, 1993.

51. Edelstein, Herb & Ramon Barquin, Planning and Designing the Data Warehouse, The data warehousing institute series from Prentice Hall PTR, New Jersey, 1997.

52. Eder J., "BIER – the Behavior Integrated Entity-Relationship Approach", 5th International Conference on Entity Relationship Approach, North-Holland, 1986.

53. Egenhofer, Max J. & Andrew U. Frank, Jeffrey P. Jackson, "A Topological Data Model for Spatial Databases", SSD, Santa Barbara, California, pp. 271-286, 1989.

54. Eick C.F. & P.C. Lockemann, "Acquisition of Terminological Knowledge Using Database Design Techniques", ACM SIGMOD, Austin, Texas, 1985.

55. Falkenberg E., "Concepts for Modeling Information", Modeling in DBMS, North-Holland, 1976.

56. Falkenberg E.D.H. & V. Kempen, "Knowledge-Based Information Analysis Support", Artificial Intelligence in Database and Information System, Guangzhou, China, 1988.

57. Farias de Souza, Marcio & Marcus Costa Sampaio, "Efficient Materialization and Use of Views in Data Warehouses", SIGMOD28(1), Philadelphia, PA, pp.78-83, 1999.

58. Francis Day, Young & Serhan Dagtas, Mitsutoshi Lino, Ashfaq Khokhar, Arif Ghafoor, "An Object-Oriented Conceptual Modeling of Video Data", ICDE, Taipei, Taiwan, pp.401-408, 1995.

59. Motschnig-Pitrik, Renate "The Semantics of Parts Versus Aggregates in Data/ Knowledge Modeling", CaiSE, Paris, France, pp.352-373, 1993.

60. Geffner, S. & Divyakant Agawal, Amr El Abbadi, Terence R. Smith, "Relative Prefix Sums: An Efficient Approach for Querying Dynamic OLAP Data Cubes", ICDE, Sydney, Australia, 1999.

61. Gelenbe, Erol & Georges Hebrail, "A Probability Model of Uncertainty in Data Bases", ICDE, Los Angeles, California, pp. 328-333, 1986.

62. Gingras, Frederic & Laks V. S. Lakshmanan, "A Multi-dimensional Language for Interoperability and OLAP", VLDB, New York City, USA, pp.134-145, 1998.

63. Goil, Sanjay and Alok Choudhary, "An Infrastructure for Scalable Parallel Multidimensional Analysis", Northwestern University, International Conference on Scientific and Statistical Database Management, July 1999.

64. Golfarelli, Matteo & Dario Maio, Stefano Rizzi, Conceptual Design of Data Warehouses from E/R Scheme, Published in the Proceeding of the Hawaii International Conference On System Sciences, January 6-9,1998.

65. Golfarelli, Matteo & Stefano Rizzi, "Methodological Framework for Data Warehouse Design", DOLAP, Bethesda, Maryland, 1998.

66. Gray, Jim & S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab, and Sub Total", Data Mining and Knowledge Discovery 1, Tucson, Arizona, 1997.

67. Greenfield, Larry "Infrastructure Technology Vendors", LGI System Incorporated, 1999.

68. Grosz G. & C. Rolland, "Using Artificial Intelligence Techniques to Formalize the Information System Design Process", Database and Expert System Application, DEXA 90, Vienna, Austria, 1990.

69. Gupta H. and I. Mumick, "Selection of Views to Materialize in a Data Warehouse", Proceedings of the International Conference on Database Theory (ICDT), Greece, January 1997.

70. Gupta, A. & V. Harinarayan, D. Quass, "Aggregate-Query Proceeding in Data Warehousing Environments", Proceedings of the 21st VLDB Conference, Zurich, Switzerland, Sept. 1995.

71. Gupta, Amarnath & Terry E. Weymouth, Ramesh Jain, "An Extended Object-Oriented Data Model for Large Image Bases", SSD, Zürich, Switzerland, pp. 45-61, 1991.

72. Gyssens, Marc & Jan Paredaens, Dirk Van Gucht, "A Grammar-Based Approach Towards Unifying Hierarchical Data Models", SIGMOD Conference, Portland, Oregon, pp. 263-272, 1989.

73. Gupta H. and I.S.Mumick, "Incremental Maintenance of Aggregate and Outerjoin Expressions", Technical Report, Standford University, Stanford, CA, 1999.

74. Hammer H., McLead D., "The Semantic Data Modeling: A Modeling Mechanism for Database Applications", Proc. ACM SIGMOD Conference, Austin, Texas, 1978.

75. Han, Jiawei "OLAP Mining: Integration of OLAP with Data Mining", DS-7, Leysin, Switzerland, 1997.

76. Hanson, Joseph H. & Mary Jane Willshire, "Modeling a Faster Data Warehouse", IDEAS, Montreal, Canada, pp.260-268, 1997.

77. Hou, Wen-Chi "A Framework for Statistical Data Mining On Summary Tables", Southern Illinois University at Carbondale, International Conference on Scientific and Statistical Database Management, July 1999.

78. Hsu, Cheng & Alvaro Perry, M'hamed Bouziane, Waiman Cheung, "TSER: A Data Modeling System Using the Two-Stage Entity-Relationship Approach", ER, New York, USA, pp.497-514, 1987.

79. Hurtado, Carlos A. & Alberto O. Mendelzon, Alejandro A. Vaisman, "Maintaining Data Cubes Under Dimension Updates", ICDE, Sydney, Australia, 1999.

80. Hurtado, Carlos A. & Alberto O. Mendelzon, Alejandro A. Vaisman, "Updating OLAP Dimensions", DOLAP, Kansas City, Missouri, USA, pp.60-66, 1999.

81. Hurtado, Carlos A. & Alberto O. Mendelzon, Alejandro A. Vaisman, "Multidimensional Data Modeling for Complex Data", ICDE, Sydney, Australia, 1999.

82. Huyn, N. "Efficient Self-Maintenance of Materialized Views", Technical Report, Standford University, Stanford, CA, 1996.

83. Huyn, N. "Efficient View Self-Maintenance", Proceedings of the ACM Workshop on Materialized Views: Techniques and Applications, Montreal, Canada, June 1996.

84. Huyn, N. "Exploiting Dependencies to Enhance View Self-Maintainability" Technical Report, Standford University, Stanford, CA,1997.

85. Huyn, N. "Multiple-View Self-Maintenance in Data Warehousing Environments", Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997.

86. J.A. Bubenko JR. & B. Wangler, "Research Directions in Conceptual Specification Development", Conceptual Modeling, Database, and Case, pp.49-68, 1992.

87. Jagadish H.V. & Kapitskaia O., Ng R.T. & Srivastava D., "Multi-Dimensional Substring Selectivity Estimation", 25th International Conference on Very Large Data Bases, Scotland, September 1999.

88. Jeusfeld, Manfred A. & Christoph Quix, Matthias Jarke, "Design and Analysis of Quality Information for Data Warehouses", ER, Singapore, pp.349-362, 1998.

89. Johannesson, B. M. & C. Sundblad, "View Integration – A Knowledge Problem, SYSLAB", Stockholm University, Stockholm, 1987.

90. Johansson, Olof "Using an Extended ER-Model Based Data Dictionary to Automatically Generate Product Modeling Systems", ADB, Vadstena, Sweden, pp. 42-61, 1994.

91. Johnson, R. R. "Modeling Summary Data", SIGMOD Conference, Ann Arbor, Michigan, pp. 93-97, 1981.

92. Jurgens, Marcus & Hans-J. Lenz, "The R-Tree: An Improved R-tree with Materialized Data for Supporting Range Queries on OLAP Data", DEXA Workshop, Vienna, Austria, pp.186-191, 1998.

93. Kamel, Nabil & Roger King, "A Model of Data Distribution Based on Texture Analysis.", SIGMOD, Austin, Texas, pp. 319-325, 1985.

94. Kawaguchi, A. & D. Lieuwen, I. Mumick, D. Quass, K. Ross, "Concurrency Control Theory for Deferred Materialized Views", Proceeding of the Conference on Database Theory, Athens, Greece, Jan. 1997.

95. Kenan Technologies, "An Introduction to Multidimensional Database Technology".

96. Kimball, Ralph "A Dimensional Modeling Manifesto", DBMS Magazine, August 1997.

97. Klir, George J. & Ute H. St. Clair and Bo Yuan, Fuzzy Set Theory: Foundations and Applications., Prentice-Hall International, Inc., New Jersey, 1997

98. Labio, W. & J. Yang, Y. Cui, H. Garcia-Molina, and J. Widom, "Performance Issues in Incremental Warehouse Maintenance", Technical Report, Standford University, Stanford, CA, 1999.

99. Labio, W. J. & J. Wiener, H. Garcia-Molina, V. Gorelik, "Efficient Resumption of Interrupted Warehouse Loads", Technical Report, Standford University, Stanford, CA, 1998.

100. Labio, W. J. & R. Yerneni, and H. Garcia-Molina, "Shrinking the Warehouse Update Window", Proceedings of the ACM SIGMOD Conference, Philadelphia, PA, May 1999.

101. Labio, Wilburt & Dallan Quass, Brad Adelberg, "Physical Database Design for Data Warehouses", Proceedings of the International Conference on Data Engineering (ICDE), Birmingham, UK. 1997.

102. Lahlou, Youssef & Noureddine Mouaddib, "Relaxing the Instantiation Link: Towards a Content-Based Data Model for Information Retrieval", CaiSE, Crete, Greece, pp. 540-561, 1996.

103. Lazimy, Rafael "ER Model and Object-Oriented Representation for Data Management, Process Modeling, and Decision Support", ER, Toronto, Canada, pp. 129-151, 1989.

104. Lee, Kyuchul & Sukho Lee, "An Object-Oriented Approach to Data/Knowledge Modeling Based on Logic", ICDE, Los Angeles, California, pp. 11-19, 1990.

105. Lehner, W. "Modeling Large Scale OLAP Scenarios", EDBT, Valencia, Spain, March 1998.

106. Lenz, Hans-J. & Arie Shoshani, "Summarizability in OLAP and Statistical Data Bases", SSDBM, Berlin, Germany, pp.132-143, 1997.

107. Leonard,Michel & Ian Prince, "A Framework for Literate Data Modeling", CaiSE, Manchester, UK, pp. 239-256, 1992.

108. Lewerenz, Jana & Klaus-Dieter Schewe, Bernhard Thalheim, "Modeling Data Warehouses and OLAP Applications by Means of Dialogue Objects" ER, Paris, France, pp.354-368, 1999.

109. Li Chang and X. Sean Wang, George Mason University, A Data Model for Supporting On-Line Analytical Processing, CIKM, Rockville, Maryland, pp. 81-88, 1996.

110. Li J., Rotem D. & Srivastava J., "Aggregation Algorithms for Very Large Compressed Data Warehouses", 25th International Conference on Very Large Data Bases, Scotland, September 1999.

111. Lubars, M.D. & M.T. Harandi, "Intelligent Support for Software Specification and Design", IEEE Expert, USA, 1986.

112. Lyngbaek, Peter & William Kent, "A Data Modeling Methodology for the Design and Implementation of Information Systems", OODBS, Pacific Grove, California, USA, pp. 6-17, 1986.

113. Mangisengi, O. & A. Min Tjoa, "A Multidimensional Modeling Approach for OLAP within the Framework on the Relational Model Based on Quotient Relations", DOLAP, Bethesda, Maryland, 1998.

114. Mani, D. R. & Jame Drew, Andrew Betz, Piew Datta, "Statistics and Data Mining Techniques for Lifetime Value Modeling", KDD, San Diego, California, USA, pp.94-103, 1999.

115. Marques, Paolo & Paula Furtado, Peter Baumann, "An Efficient Strategy for Tiling Multidimensional OLAP Data Cubes", Workshop Data Mining and Data Warehousing, pp.13-24, 1998.

116. Mattos, Nelson Mendonca "The Basis for Data and Knowledge Modeling", ER, Rome, Italy, pp. 473-492, 1988.

117. McGuff, Frank "Data Modeling for Data Warehouses", Oct 1996.

118. McLeod, Dennis "On Conceptual Database Modeling" Workshop on Data Abstraction, Databases and Conceptual Modeling, pp. 161-163, 1980.

119. Meleod D. & Hammer M., "Database Description with SDM: Semantic Database Model", ACM TODS 6(3), Arizona, USA, Sept 1981.

120. Miyamoto, Isao "Hierarchical Performance Analysis Models for Data Base Systems", VLDB, Framingham, Massachusetts, pp.322-352, 1975.

121. Mohan, Narendra "DWMS: Data Warehouse Management System", VLDB, Bombay, India, 1996.

122. Motschnig-Pitrik, Renate "The Semantics of Parts Versus Aggregates in Data/ Knowledge Modeling", CaiSE, Paris, France, pp.352-373, 1993.

123. Muck T. & Vinek G., "Modeling Dynamic Constraints in Database", Expert Systems and Knowledge Representation, Proc. 1st Workshop on Expert System, 1984.

124. Mumick, I. & D. Quass, B. Mumick, "Maintenance of Data Cubes and Summary Tables in a Warehouse", Proceedings of the ACM SIGMOD Conference, Tuscon, Arizona, May 1997.

125. Mumick, Inderpal Singh & Dallan Quass, Barinderpal Singh Mumick, "Maintenance of Data Cubes and Summary Tables in a Warehouse", SIGMOD Conference, Tucson, Arizona, pp.417-217, 1997.

126. Muntz, Alice H. & Christian T. Ramiller, "A Requirement-Based Approach to Data Modeling and Re-engineering", VLDB, Santiago de Chile, Chile, pp.643-654, 1994.

127. Mylopoulos J., Bernstein P.A., Wong H.K.T, "A Language Facility for Designing Database Intensive Applications", ACM TODS Arizona, USA, Vol.15, No.2, 1980.

128. Mylopoulos, J. "Conceptual Modeling and Telos", Conceptual Modeling, Database, and Case, pp.49-68, 1992.

129. Nah, Yunmook & Sukho Lee, "Two-level Modeling Schemes for Temporal-Spatial Multimedia Data Representation.", DEXA, Valencia, Spain, pp. 102-107, 1992.

130. Nicolle, Christophe & Djamal Benslimane, Kokou Yetongnon, "Multi-Data Models Translations in Interoperable Information Systems", CaiSE, Crete, Greece, pp. 176-192, 1996.

131. Nijssen and Halpin, "Conceptual Schema and Relational Database Design – A Fact-Oriented Approach", Prentice-Hall, 1989.

132. Oracle Corporation, Oracle Express Database Design and Cotrol Manual, Oracle Corporation Publisher, 1995.

133. Oren, Ole & Frode Aschim, "Statistic for the Usage of a Conceptual Data Model as a Basis for Logical Data Base Design", VLDB, Rio de Janeiro, Brazil, pp.140-145, 1979.

134. Ozkarahan, Esen "Database Management Concept, Design, and Practice", Prentice Hall, New Jersey, 1990.

135. P. O'Neil & D. Quass, "Improved Query Performance with Variant Indexes", Proceedings of the ACM SIGMOD Conference, Tuscon, Arizona, May 1997.

136. Pedersen B.T., Jensen C. S. & Dyreson C. E., "Extending Practical Pre-Aggregation in On-Line Analytical Processing", 25th International Conference on Very Large Data Bases, Scotland, September 1999.

137. Pedersen, Torben Bach "Supporting Imprecision in Multidimensional Databases Using Granularities", Aalborg University, Denmark, International Conference on Scientific and Statistical Database Management, July 1999.

138. Pedersen, Torben Bach & Christian S. Jensen, "Multidimensional Data Modeling for Complex Data", ICDE, Sydney, Australia, pp.336-345, 1999.

139. Pietri F. & P.P. Puncello, "ASPIS: A Knowledge-Based Environment for Software Development (ESPRIT)", Proceeding ESPRIT'87, Brussels, Belgium, 1987.

140. Pokorny, Jaroslav "Conceptual Modeling of Statistical Data", DEXA Workshop, Zurich, Switzerland, pp.377-382, 1996.

141. Poosala, Viswanath & Bell Laboratories, Venkatesh, Ganti, "Fast Approximate Answers to Aggregate Queries on a Data Cube", University of Wisconsin, International Conference on Scientific and Statistical Database Management, July 1999.

142. Pourabbas, Elaheh & Maurizio Rafanelli, "Characterization of Hierarchies and Some Operators in OLAP Environment", DOLAP, Kansas City, Missouri, USA, pp.54-59, 1999.

143. Punchello P.P. & P. Torrigiani, "ASPIS: A Knowledge-based CASE Environment", March, pp. 58-65, 1988.

144. Qadah, Ghassan Z. "An Inference Model and a Tree-Structured Multicomputer System for Large Data-Intensive Logic-Bases", IWDM, Tokyo, Japan, pp. 503-516, 1987.

145. Quass D. and J. Widom, "On-Line Warehouse View Maintenance for Batch Updates", Proceedings of the ACM SIGMOD Conference, Tuscon, Arizona, May 1997.

146. Quass, D. "Maintenance Expressions for Views with Aggregation", Proceedings of the ACM Workshop on Materialized Views: Techniques and Applications, Montreal, Canada, June 1996.

147. Quass, D. & A. Gupta, I. Mumick, and J. Widom, "Making Views Self-Maintainable for Data Warehousing", Proceedings of the Conference on Parallel and Distributed Information Systems, Miami Beach, FL, December 1996.

148. Raden, Neil "Modeling the Data Warehouse", Information Week, Jan. 1996.

149. Ramakrishnan, R. & K. A. Ross, D. Srivastava, S. Sudarshan, "Efficient Incremental Evaluation of Queries with Aggregation", International Symposium on Logic Programming, Nov. 1994.

150. Ramsak, Frank & Volker Markl, Rudolf Bayer, "Physical Data Modeling for Multidimensional Access Methods", Grundlagen von Datenbanken, pp.97-101, 1999.

151. Rauh, Otto "Some Rules for Handling Derivable Data in Conceptual Data Modeling", DEXA, Valencia, Spain, pp. 500-505, 1992.

152. Rea, Alan Queen's University of Belfast, Data Mining - Introduction and Concept, Parallel Computer Centre, December 1995.

153. Read, Robert L. & Donald S. Fussell, Abraham Silberschatz, "A Multi-Resolution Relational Data Model", VLDB, Vancouver, British Columbia, Canada, pp.139-150, 1992.

154. Rolland, C. & C. Canvet, "Trends and Perspectives in Conceptual Modeling", Conceptual Modeling, Databases, and Cases, John Wiley and Sons Inc., p.27-48, 1992.

155. Ross D.T., Schoman K.E., "Structured Analysis for Requirements Definition", IEEE Trans. SE(3/1), pp.1-65, 1977.

156. Ross, K. A. & D. Srivastava, D. Chatziantoniou, "Complex Aggregation at Multiple Granularities", International Conference on Extending Database Technology, March 1998.

157. Ross, K. A. & D. Srivastava, P. J. Stuckey, S. Sudarshan, "Foundations of Aggregation Constraints", Theoretical Computer Science, pp. 149-179, Feb 1998.

158. Roussopoulos, Nick "Materialized Views and Data Warehouses", KRDB, Athens, Greece, pp.12.1-12.6, 1997.

159. Roussopoulos, Nick "Materialized Views and Data Warehouses", SIGMOD Record 27(1), Seattle, Washington, pp.21-26, 1998.

160. Rumbaugh J., "Object-Oriented Modeling and Design", Prentice-Hall, 1991.

161. Schiel, Ulrich "An Abstract Introduction to the Temporal-Hierarchic Data Model (THM)", VLDB, Florence, Italy, pp.322-330, 1983.

162. Senko M., "DIAM as a Detail Example of ANSI/SPARC Architecture – In Modeling in Database System", Nijssen G., North-Holland, 1976.

163. Seo, Dongsu & Pericles Loucopoulos, "Formalisation of Data and Process Model Reuse Using Hierarchic Data Types", CaiSE, Utrecht, The Netherlands, pp.256-268, 1994.

164. Sevcik, Kenneth C. "Data Base System Performance Prediction Using an Analytical Model", VLDB, Cannes, France, pp.182-198, 1981.

165. Shao, Shin-Chung "Multivariate and Multidimensional OLAP", EDBT, Valencia, Spain, pp.120-134, 1998.

166. Shasha, Dennis "Netbook – a Data Model to Support Knowledge Exploration", VLDB, Stockholm, Sweden, pp.418-425, 1985.

167. Shipman D.W., "The Functional Data Model and the Language DAPLEX", ACM TODS 6(1), Arizona, USA, 1981.

168. Shoshani, Arie "OLAP and Statistical Databases: Similarities and Differences", PODS, pp.185-196, 1997.

169. Shoval, Peretz & Sagit Shiran, "Entity-Relationship and Object-Oriented Data Modeling - an Experimental Comparison of Design Quality", DKE 21, pp.295-315, 1997.

170. Shukla, Amit & Prasad Deshpande, Jeffrey F. Naughton, Karthikeyan Ramaswamy, "Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies", VLDB, Bombay, India, 1996.

171. Silvon, "Defining Data Warehousing: What is it and who need it?", A Silvon Software, Inc. White Paper, Westmont, IL.

172. Singh, Harry S., Data Warehousing Concepts, Technologies, Implementations, *and Management*, Prentice Hall PTR, New Jersey, 1998.

173. Smith J.M. & Smith D.C.P., "Database Abstractions: Aggregation and Generalization", ACM Trans. On Database System (TODS) Vol.2 No.2, Arizona, USA, pp.105-133, 1997.

174. Spaccapietra S. & C. Parent, "View Integration: A Step Forward in Solving Structural Conflicts", Ecole Polytechnique Federale, Switzerland, 1990.

175. Stanoi, Ioana & Divyakant Agrawal, Amr El Abbadi, "Modeling and Maintaining Multi-View Data Warehouses", ER, Paris, France, pp.161-175, 1999.

176. Stanoi, Loana & Divyakant Agrawal, Amr El Abbadi, "Modeling and Maintaining Muli-View Data Warehouses", ER, Paris, France, pp. 161-175, 1999.

177. Su S.Y.W., "A Semantic Association Model for Corporate and Scientific Statistical Databases", Inf. Science 29, pp.151-199, 1982.

178. Sutton, David R. P. & J. H. King, "Integration of Model Logic and the Functional Data Model", BNCOD, Aberdeen, Scotland, pp. 156-174, 1992.

179. Tauzovich B. "An Expert System for Conceptual Data Modeling", 8th Conference on ER Approach, Toronto, Canada, pp.329-344, 1989.

180. Theodoratos, Dimitri & Timos K.Sellis, "Designing Data Warehouses", DKE31 (3), pp.279-301, 1999.

181. Theodoulidis C. & Loucopoulos P., Wangler B., "The Entity Relationship Time Model and the Conceptual Rule Language", Proc. 10th International Conference on Entity Relationship Approach, San Mateo, California, Oct. 1991.

182. Theodoulidis, C. & B. Wangler, P. Loucopoulos, "The Entity-Relationship-Time Model", Conceptual Modeling, Database, and Case, pp.49-68, 1992.

183. Tsubaki, Masaaki "Multi-Level Data Model in DPLS – Database, Dynamic Program Control & Open-Ended Pol Support", VLDB, Framingham, Massachusetts, pp.538—539, 1975.

184. Wang K., Zhou S. & Liew S.C., "Building Hierarchical Classifiers Using Class Proximity", 25th International Conference on Very Large Data Bases, Scotland, September 1999.

185. Wang, Bing "Toward a Unified Data Model for Large Hypermedia Applications", DEXA, Toulouse, France, pp. 142-151, 1997.

186. Welzer, Tatjana & Johann Eder, "Meta Data Model for Database Design", DEXA, Prague, Czech Republic, pp. 677-680, 1993.

187. Wietek, Frank "Modeling Multidimensional Data in a Dataflow-Based Visual Data Analysis Environment", CaiSE, Heidelberg, Germany, pp.149-163, 1999.

188. Wohed R. "Diagnosis of Conceptual Schemas", Artificial Intelligence in Database and Information System, Guangzhou, China, 1988.

189. Y. Cui and J. Widom, "Storing Auxiliary Data for Efficient View Maintenance and Lineage Tracing", Technical Report, Standford University, Stanford, CA, 1999.

190. Yang J. and J. Widom, "Maintaining Temporal Views Over Non-Temporal Information Sources for Data Warehousing", Proceedings of the 6th International Conference on Extending Database Technology, Valencia, Spain, March 1998.

191. Yang J. and J. Widom, "Making Temporal Views Self-Maintainable for Data Warehousing", Proceeding of the 7th International Conference on Extending Database Technology, Konstanz, Germany, March 2000.

192. Zhang, Chuan & Jian Yang, "Genetic Algorithm for Materialized View Selection in Data Warehouse Environments", DaWak, pp.116-124, 1999.

193. Zhang, Chuan & Jian Yang, "Materialized View Evolution Support in Data Warehouse Environment", EDBT, Konstanz, Germany, pp.293-307, 1999.

194. Zhang, Xin & Elke A. Rundensteiner, "Data Warehouse Maintenance under Concurrent Schema and Data Updates", ICDE, Sydney, Australia, pp.253, 1999.

195. Zhao, Yihong & Prasad Deshpande, Jeffrey F. Naughton, "An Array-Based Algorithm for Simultaneous Multidimensional Aggregations", SIGMOD, Tucson, Arizona, 1997.

196. Zhuge, Y. & H. Garcia-Molina, "Consistency Algorithms for Multi-Source Warehouse View Maintenance", Journal of Distributed and Parallel Databases, vol. 6, pp. 7-40, Jan. 1998.

197. Zhuge, Y. & H. Garcia-Molina, J. L. Wiener, "The Strobe Algorithms for Multi-Source Warehouse Consistency", Proceeding of the Conference on Parallel and Distributed Information Systems, Miami Beach, FL, Dec. 1996.

198. Zhuge, Y. & H. Garcia-Molina, J. Hammer, and J. Widom, "View Maintenance in a Warehousing Environment", Proceeding of the ACM SIGMOD Conference, San Jose, California, May 1995.

199. Zhuge, Y. & H. Garcia-Molina, "Graph Structure Views and Their Incremental Maintenance", Proceeding of the International Conference on Data Engineering, Orando, FL, Feb. 1998.

200. Zhuge, Y. & J.L. Wiener and H. Garcia-Molina, "Multiple View Consistency for Data Warehousing", Proceeding of the International Conference Data Engineering, Binghamton, UK, April 1997.